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, 30.08.2025

Eigenständigkeitserklärung

Wir bestätigen hiermit, dass diese Projektdokumentation von uns verfasst wurde und auf unserer eigenen Arbeit basiert, sofern nicht anders angegeben. Die Arbeiten anderer Personen wurden nur unter angemessener Angabe der Quelle verwendet. Alle Zitate und wörtlichen Auszüge wurden kenntlich gemacht, und alle Informationsquellen, einschließlich Grafiken und Datensätze, sind ausdrücklich angegeben.

Pforzheim, 30.08.2025 Silja-Marie Fischer und Julia Krauß

Inhaltsverzeichnis

Hinweis zum Sprachgebrauch

In dieser Dokumentation wird aus Gründen der Lesbarkeit und Verständlichkeit auf geschlechtsneutrale Schreibweisen mit explizitem Genderstern oder ähnlichen Formen verzichtet. Begriffe wie „Entwickler“ oder „Nutzer“ gelten im Text jeweils geschlechtsunabhängig und schließen alle Geschlechter ein.

Abstract

In den letzten Jahren gewinnt Mixed Reality (MR) zunehmend an Bedeutung, besonders für Bildungs- und Unterhaltungsanwendungen. In dieser Arbeit wird ein Mixed-Reality-Prototyp zur virtuellen Tierbeobachtung auf der Meta Quest 3 entwickelt. Die Anwendung ermöglicht es, virtuelle Tiere in der realen Umgebung zu beobachten und zu fotografieren. Die Architektur folgt dem MVVM-3D-Muster, das Daten, Logik und Benutzeroberfläche klar voneinander trennt und gleichzeitig in der Library komplexere Berechnungen durchführt. Ein zentraler Service sorgt für die Verbindung zwischen Anwendung und Datenbasis. Der Prototyp zeigt, wie MR-Anwendungen mit realistischen Animationen, intuitiver Bedienung und raumbezogener Interaktion umgesetzt werden können.

Abkürzungsverzeichnis

API ..... Application Programming Interface
AR ..... Augmented Reality
B2B ..... Business-to-Business
B2C ..... Business-to-Consumer
CSV ..... Comma-Separated Values
ECS ..... Entity Component System
HMD ..... Head-Mounted Display
HTTP ..... Hypertext Transfer Protocol
LCD ..... Liquid Crystal Display
LINQ ..... Language Integrated Query
MIT ..... Master Advanced Information Technology
MR ..... Mixed Reality
MRUK ..... Mixed Reality Utility Kit
MVC ..... Model-View-Controller
MVVM ..... Model-View-ViewModel
NDK ..... Native Development Kit
OLED ..... Organic Light-Emitting Diode
OpenJDK ..... Open Java Development Kit
SDK ..... Software Development Kit
SQL ..... Structured Query Language
UI ..... User Interface
VR ..... Virtual Reality
WWF ..... World Wide Fund of Nature
XR ..... Extended Reality

1 Einleitung

1.1 Motivation

Die immer stärkere Digitalisierung der Welt und häufige Nutzung mobiler Geräte haben das Lernen und die Wahrnehmung von Informationen stark beeinflusst. 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 (Max-Planck-Institut für Bildungsforschung, 2025). Aktuelle Forschungen zeigen allerdings auch, dass interaktive Lernmethoden, besonders Gamifikation und Extended Reality (XR), die Motivation der Nutzer steigern und die Aufmerksamkeit stärken. Gamifizierte Elemente wie Herausforderungen, Belohnungen und interaktive Inhalte fördern nicht nur das Lernen, sondern erhöhen auch flexibles Denken 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 (Ratinho & Martins, 2023).

Ein konkretes Anwendungsbeispiel zeigt, wie VR-Umgebungen erfolgreich in der Lehre eingesetzt werden können. In der Forstwirtschaft werden beispielsweise virtuelle Abbildungen realer Waldbestände erstellt, durch die Waldwachstum, forstliche Prozesse und die Auswirkungen von Umweltveränderungen realitätsnah dargestellt werden. Diese Umgebungen dienen nicht nur der Ausbildung von Forstexperten und der Forschung, sondern bieten auch externen Partnern Zugang (Universität Freiburg, 2023). Ein zentrales Einsatzfeld ist auch die Unfallprävention. Da die Forstwirtschaft mit einem hohen Unfallrisiko verbunden ist, ermöglicht VR-Training die praxisnahe Vorbereitung auf Gefahren und erhöht die Sicherheit der Lernenden (Goebel & ORF-Wissenschaft, 2023). Vor dem Hintergrund des Klimawandels und neuer Anforderungen an Waldbesitzer ist die Weiterentwicklung von Aus- und Weiterbildung in der Forstwirtschaft notwendig. Spätestens durch die Corona-Pandemie wurde zudem deutlich, wie wichtig digitale Lehr- und Lernangebote geworden sind. VR-gestützte Lernmethoden stellen daher nicht nur eine innovative Verbindung von Didaktik und Technologie dar, sondern bieten auch einen wesentlichen Mehrwert für die Lernerfahrung und die flexible Gestaltung von Aus- und Weiterbildung (Fell, 2021).

Mixed Reality kann außerdem das psychische Wohlbefinden 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, vor allem 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 (Wilczyńska et al., 2024).

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. Dies kann dazu beitragen, das Bewusstsein für den Naturschutz zu stärken und das Interesse an der Tierwelt zu wecken (Rambach et al., 2020).

Besonders in einer zunehmend digitalisierten Gesellschaft kann die virtuelle Naturerfahrung eine wertvolle Verbindung zwischen den Nutzern und der Umwelt erzeugen. Die Anwendung von MR zur Förderung des Naturschutzes und des psychischen Wohlbefindens stellt daher eine vielversprechende Lösung dar, um das Lernen zu optimieren und das Bewusstsein für die Natur zu stärken.

1.2 Problemdefinition

Digitale Anwendungen zur Tierbeobachtung und Naturerfahrung sind in den letzten Jahren zunehmend verfügbar. Häufig bieten diese jedoch kein umfassendes digitales und immersives Lernerlebnis. Viele bestehende Spiele basieren ausschließlich auf Virtual Reality. Sie bieten kaum Möglichkeiten zur Bewegung im realen Raum und erlauben nur begrenzte Interaktion mit der Umgebung oder den dargestellten Tieren. Anwendungen auf Basis von Mixed Reality, die virtuelle Tiere in die reale Umgebung einbinden, gibt es bisher kaum.

Zudem wird der edukative Mehrwert in den meisten Lösungen vernachlässigt. Gamifizierte Elemente wie das Sammeln von Beobachtungen in einem digitalen Tagebuch oder interaktive Aufgaben, die das Wissen über Tiere und Lebensräume vertiefen, sind kaum integriert. Dadurch fehlt es an Anreizen, die eine langfristige Motivation und nachhaltiges Lernen unterstützen.

Insgesamt zeigt sich ein deutlicher Bedarf im Bereich Mixed-Reality-Anwendungen. Es gibt fast keine, die Tierbeobachtung, Fotografie, Interaktion und Wissensvermittlung sinnvoll miteinander kombinieren. Neue Anwendungen könnten diesen Bedarf decken, indem sie reale und virtuelle Inhalte kombinieren, spielerisches Erleben mit Bildung verbinden und Nutzer langfristig motivieren.

1.3 Projektziel

Das Hauptziel dieses Projekts ist also die Entwicklung einer prototypischen 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, zum Beispiel weil sie in abgelegenen 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.

1.4 Aufbau der Dokumentation

Die Dokumentation ist in sieben Überkapitel unterteilt. Diese umfassen den gesamten Entwicklungsprozess der Mixed-Reality-Anwendung.

Die Einleitung in Kapitel 1 zeigt die Motivation und Potentiale für die Erstellung der Anwendung zur virtuellen Tierbeobachtung auf. Es werden aktuell bestehende Probleme definiert und daraus ein Projektziel erarbeitet.

In Kapitel 2 wird der Stand der Technik vorgestellt. Hierbei werden die aktuellen Grundlagen und Technologien im Bereich der Extended Reality (XR) erläutert. Dabei werden die Begriffe Virtual Reality (VR), Augmented Reality (AR) und Mixed Reality (MR) definiert und voneinander abgegrenzt. Es wird beschrieben, wie diese Technologien zurzeit in XR-Spielen und anderen Anwendungsbereichen zur Tierbeobachtung eingesetzt werden.

Kapitel 3 stellt die aus dem Projektziel herausgearbeiteten Anforderungen an die MR-Anwendung vor. Anhand dieser können dann ein Konzept erarbeitet und die einzelnen Funktionen in die Anwendung implementiert werden.

In Kapitel 4 wird die Technologieauswahl beschrieben. Es wird erläutert, welche Hardware, Entwicklungsumgebung, Programmiersprache und Architekturmodelle für die Umsetzung des Projekts verwendet werden. Es wird auf die Überlegungen eingegangen, die diese Entscheidungen beeinflussen.

Kapitel 5 beschreibt das Konzept des Prototyps. Hier wird der Aufbau des Prototyps erläutert. Dies umfasst die Struktur der Datenbank und die Planung des Backends und Frontends der Anwendung.

Die Implementierung und Prototypenentwicklung wird in Kapitel 6 behandelt. Es wird beschrieben, wie die einzelnen Konzeptionierungen zur Erstellung der Anwendung umgesetzt werden.

Abschließend gibt Kapitel 7 eine Zusammenfassung der Ergebnisse des Projekts und einen Ausblick auf mögliche Weiterentwicklungen und zukünftige Anwendungsmöglichkeiten.

2 Stand der Technik

2.1 Extended Reality

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

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

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

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

Auch wenn ein wachsendes Interesse im XR-Bereich beobachtet werden kann, kann von keiner belegbaren Marktreife entsprechender Produkte gesprochen werden. Jedoch bieten Eigenschaften wie Interaktivität, Medienreichtum oder auch Immersion einige Potenziale, wie sich insbesondere in der Werbeindustrie zeigt. Die genannten Eigenschaften fördern die Zugänglichkeit zu bestimmten Produkten oder Themen und können so die Kaufeigenschaft und Haltung gegenüber Werbung beeinflussen (Barta et al., 2025; Mehler-Bicher & Steiger, 2021).

[1] Cardboards sind Pappvorrichtungen, in die ein Smartphone geschoben werden kann, welches dann ein Bild mit einem Eindruck räumlicher Tiefe erstellt.

So werden heute in Webshops 3D-Objekte-Brillen ins Gesicht projiziert (Living Mirror, als Anwendung über das Smartphone) oder geplante Küchen virtuell begutachtet (Living Architecture) (*Brille online anprobieren: Teste den virtuellen 3D-Simulator*, o. J.; *Virtuelle 3D-Küchenplanung | Der Wohnfuchs*, o. J.; Mehler-Bicher & Steiger, 2014). Für immersive Welten ist die Kategorie Living Environment relevant, denn diese beinhaltet Anwendungen, die reale Umgebungen abbilden und dabei eine Kombination mehrerer sensorischer Reize ergänzen. Das können 3D-Objekte oder auch Audiosequenzen sein (Mehler-Bicher & Steiger, 2014).

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

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

Technologisch herausfordernde Aspekte umfassen sowohl Software- als auch Hardware-Limitierungen, aber auch Themen wie Tracking und Interaktion. Der Bereich Interaktion umfasst nicht nur den Übergang unterschiedlicher Welten, sondern auch das Thema „Kollaboration“. So sind die nahtlose Interaktion und Latenzen, sowie der Datenschutz und Echtzeitsynchronisation Themen, die die Entwicklung stabiler Multiplayeranwendungen beeinflussen. Allerdings zeigen gerade kollaborative virtuelle Umgebungen in einer immer digitalisierteren (Geschäfts-)Welt die Notwendigkeit für weitere Forschung und Standardisierung, wie Anwendungen wie virtuelle Meetingräume belegen (Lapschies, o. J.; Wolfenstein, 2025). Damit für solche Anwendungen die Akzeptanz steigt, werden immersive Anwendungen benötigt. Um aber eine sehr gute Immersion mit konsistenter Performance zu erschaffen, wird leistungsfähige Hardware benötigt. So ist beispielsweise die Verarbeitung hochauflösender Grafiken ohne zu überhitzen oder den Akku zu schnell zu entleeren, ein noch nicht vollständig optimiertes Feld (Molchanov, 2025; rankmagic, o. J.). Im Softwarebereich bestehen Herausforderungen darin, Anwendungen auf verschiedenen Systemen oder Browsern zur Verfügung zu stellen. Auch unterschiedliche oder proprietäre Datenformate oder Geräteschnittstellen erschweren die Interoperabilität (Lawton, 2024). Um Latenzen zu

vermeiden und die Belastung von Geräten zu reduzieren, können Berechnungen in einer Cloud ausgeführt werden. Allerdings ist dies bei schlechter oder fehlender Internetverbindung nur eingeschränkt oder nicht möglich, was zu nicht ausführbaren Anwendungen führen kann (Molchanov, 2025; rankmagic, o. J.). Um in MR oder VR einen fließenden Übergang zwischen den Welten zu ermöglichen, sind fortschrittliche Tracking- und Renderingtechnologien erforderlich, welche derzeit noch ausbaufähig sind (Marcus, 2020; rankmagic, o. J.).

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

Um diese Herausforderungen zu adressieren, stehen einige Themen im Fokus der Forschung. Ein Aspekt ist dabei die Frage, wie plattformübergreifende Inhalte entwickelt werden können, um XR-Anwendungen zugänglicher und hardwareunabhängig zu gestalten, was wiederum die

[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).

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

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

2.2 Head-Mounted Displays

2.2.1 Begriff und Bedeutung

Head-Mounted Displays (HMDs) sind visuelle Ausgabegeräte, die direkt am Kopf getragen werden und virtuelle Projektionen unmittelbar vor den Augen des Benutzers erzeugen. Sie ermöglichen, das Sichtfeld vollständig zu erfassen und Inhalte mit Kopfbewegungen abzustimmen. Dadurch entsteht eine 360°-Ansicht. So finden HMDs in zahlreichen Bereichen wie Gaming, Ausbildung, Industrie und Marketing Anwendung (Seydel, o. J.).

Head-Mounted Displays lassen sich in drei Hauptarten unterteilen. Je nach Art können sie die reale Welt mit virtuellen Objekten erweitern oder den Anwender komplett in eine virtuelle Umgebung eintauchen lassen. See-Through-HMDs sind Brillen mit einem durchsichtigen Display, durch das die reale Umgebung vollständig wahrgenommen werden kann. Zusätzlich können digitale Objekte wie Hologramme oder Webseiten ins Sichtfeld eingeblendet werden, wodurch sie sich für Augmented Reality eignen. Look-Around-HMDs besitzen zwar kein durchsichtiges Display, sind jedoch so positioniert, dass nur ein Teil des Sichtfeldes beansprucht wird. So bleibt die reale Welt weiterhin sichtbar und kann ebenfalls durch digitale Inhalte ergänzt werden. Beide Arten ermöglichen also eine Erweiterung der Realität. Im Gegensatz dazu isolieren Non-See-Through-HMDs den Nutzer vollständig von seiner Umgebung, indem sie mit einem geschlossenen Display vor beiden Augen ausschließlich virtuelle Inhalte darstellen. Dadurch wird die reale Welt ausgeblendet und der Anwender taucht in eine

vollständig neue, virtuelle Realität ein. Diese Geräte können durch Eingabegeräte wie Controller oder auch durch Gestenerkennung erweitert werden (Seydel, o. J.).

2.2.2 Technische Funktionsweise

Die Funktionsweise von Head-Mounted Displays basiert auf mehreren Komponenten und Technologien, die zusammenarbeiten, um ein immersives Erlebnis zu schaffen. Zu den typischen Bauteilen moderner HMDs gehören unter anderem Displays, Linsen, Kameras, Sensoren, Head-Tracker, Controller, Prozessor, Speicher, Batterie, Gehäuse, Anschlüsse sowie Projektoren (sogenannte Pico-Projektoren) und weiteres Zubehör (IMARC Group, o. J.).

Displays und Linsen HMDs verwenden kleine, hochauflösende Displays, die sich direkt vor den Augen des Nutzers befinden. Das zentrale Anzeigeelement wird als Panel bezeichnet und kann technisch unterschiedlich aufgebaut sein. Üblich sind OLED-Displays (organic light-emitting diode), die selbst Licht erzeugen, oder LCDs (liquid crystal display), die von hinten durch LED-Backlights beleuchtet werden, da sie selbst kein Licht abstrahlen. Um die visuelle Darstellung realistisch zu gestalten, sind diese Displays in der Regel mit speziellen Linsen ausgestattet. HMDs verwenden unterschiedliche Linsenarten, um die Bilddarstellung vor den Augen des Nutzers zu optimieren. Einfache Linsen sind klassische Lupen, die das Bild vergrößern und ein virtuelles Bild in angemessener Distanz erzeugen. Fresnel-Linsen sind dünn und leicht, können jedoch Beugungsartefakte und chromatische Aberrationen verursachen. Pancake-Linsen falten den Lichtweg mehrfach, wodurch kompakte Bauformen möglich sind, allerdings auf Kosten von Helligkeit und Kontrast. Mikrolinsen-Arrays nutzen die Krümmung von Panel und Linse, um sehr große Sichtfelder von über 180° bei hoher Bildqualität zu ermöglichen (Wölfel, 2023). Durch die stereoskopische Darstellung, bei der für jedes Auge ein leicht versetztes Bild angezeigt wird, wird ein Eindruck von Tiefe und Raum erzeugt. Dieser Effekt verstärkt das Gefühl, sich tatsächlich in einer anderen Umgebung zu befinden (Erl & Danneberg, 2025).

Tracking und Sensorik Eine der zentralen Technologien in HMDs ist das Tracking, das die Bewegungen des Nutzers in Echtzeit erfasst. Hierfür kommen verschiedene Sensoren zum Einsatz. Zur Erfassung von Merkmalen wie Position, Richtung, Bewegung oder Druck stehen unterschiedliche Verfahren zur Verfügung. Dazu zählen optische Ansätze mit RGB- oder Tiefenkameras, Inertialsysteme wie Beschleunigungssensoren und Gyroskope sowie Laufzeitverfahren, die beispielsweise auf Ultraschall oder GPS basieren (Wölfel, 2023).

Beim **Inside-Out-Tracking** sind die Kameras und Sensoren direkt im Headset integriert. Diese erfassen die Umgebung des Nutzers und berechnen daraus die Position und Orientierung des Geräts im Raum. Ein großer Vorteil dieser Methode ist, dass keine zusätzliche externe Hardware benötigt wird, wodurch das System flexibel einsetzbar und einfacher zu installieren ist. Nutzer können sich damit freier bewegen, was insbesondere bei autarken HMDs von Vorteil ist (William, 2023).

Das **Outside-In-Tracking** funktioniert dagegen mit externen Sensoren oder Basisstationen, die die Bewegungen des Headsets und der Controller im Raum erfassen. Diese Methode zeichnet sich durch eine sehr hohe Präzision und geringe Latenz aus. Allerdings ist sie weniger mobil, da die Sensoren im Raum fest installiert werden müssen und somit ein größerer Aufwand beim Aufbau entsteht (William, 2023).

In immersiven VR-Systemen ist das **Hand- und Fingertracking** entscheidend, um eine natürliche Interaktion mit virtuellen Objekten zu ermöglichen. Die menschliche Hand verfügt über eine hohe Bewegungsfreiheit, die sowohl grob- als auch feinmotorische Bewegungen erlaubt. Für VR-Anwendungen werden neben der Position und Orientierung der Hand auch die Bewegungen der einzelnen Finger erfasst. Zur Handverfolgung kommen unterschiedliche Ansätze zum Einsatz, darunter Controller, Datenhandschuhe, optische Systeme für die Freihandinteraktion oder Elektromyographie. Bei optischen Systemen werden Kameras verwendet, um die Hände im Raum zu erkennen und die Fingerbewegungen direkt zu verfolgen. Elektromyographie misst die elektrischen Signale, die von den Muskeln der Hand erzeugt werden, und erlaubt so eine genaue Bestimmung der Fingerbewegungen auch bei verdeckten oder komplexen Gesten (Wölfel, 2023).

Einige fortschrittliche HMDs integrieren **Eye-Tracking**, auch Blickverfolgung genannt. Diese Technologie erfasst die Augenstellung und Blickrichtung des Nutzers. Dabei wird der sogenannte Blickpunkt ermittelt, also der 3D-Punkt, auf den beide Augen fixiert sind. Die reine Augenerfassung misst die Position der Augen relativ zum Kopf misst. Die Blickerfassung bestimmt dagegen, wohin der Nutzer tatsächlich schaut. In modernen HMDs erfolgt Eye-Tracking fast ausschließlich optisch. Kameras, die im Headset integriert sind, verfolgen die Augen unabhängig von der Kopfbewegung (Wölfel, 2023).

Gesichtstracking ist ebenso möglich. Es werden nonverbale Signale wie Mimik erfasst. Obere Gesichtspartien werden z.B. über Druck-Sensoren erfasst, untere Gesichtspartien über Kameras und teilweise akustische Signale. Alle Daten werden zu einem Modell zusammengeführt, das die Mimik auf Avatare übertragen und eine realistische Interaktion in der virtuellen Umgebung ermöglichen kann (Wölfel, 2023).

Passthrough-Technologie Ein weiteres Merkmal moderner MR-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 (Carter, 2024).

Audiotechnologie und Haptisches Feedback Moderne HMDs verfügen häufig über integrierte Audiotechnologien, die räumliches Audio ermöglichen. Dieses erlaubt es den Nutzern, die Richtung und Entfernung von Geräuschen in der virtuellen Umgebung realistisch wahrzunehmen. Zusätzlich haben viele HMDs haptisches Feedback, um das Gefühl der

Immersion weiter zu verstärken. Es kann durch Vibrationen, Kräfte oder andere taktile Reize erfolgen (Wölfel, 2023).

2.2.3 Verfügbare Geräte

Die Entwicklung der VR-Brillen entstand 1920 mit ersten Vorläufern in Form von Polarisationsbrillen. Das erste HMD, das sogenannte „Sword of Damocles“, wurde 1968 entwickelt und war so schwer, dass es an der Decke befestigt werden musste. Erst 1990 entstanden die ersten industriell genutzten Brillen, die Informationen in das Sichtfeld einblenden konnten. Allerdings waren diese immer noch teuer, schwer und technisch limitiert, sodass sie noch nicht für einen Massenmarkt zur Verfügung standen (Barff, o. J.; Manager, 2021). Erst 2012 wurden mit der Oculus Rift VR-Brillen für den Massenmarkt relevant. 2013 zog Google mit der Google Glass nach, scheiterten jedoch an der fehlenden gesellschaftlichen Akzeptanz. Zudem boten Cardboards preiswerte Alternativen (Zobel et al., 2018).

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:

Die **Meta Quest 3** gilt aktuell als eine der beliebtesten Standalone-Brillen. Sie überzeugt durch ein verbessertes Passthrough in Farbe, was Mixed-Reality-Anwendungen deutlich realistischer macht. Der integrierte Tiefensensor sorgt für präzise Raumerfassung. Sie kombiniert ein leichtes Design mit einem attraktiven Preis-Leistungs-Verhältnis und ist daher für den Massenmarkt von zentraler Bedeutung. Zusätzlich hat Meta mit der **Meta Quest 3s** eine günstigere Variante mit einigen Abstrichen veröffentlicht, die den Zugang zu VR für eine breitere Zielgruppe erleichtert (Erl & Danneberg, 2025).

Die mit der **Vision Pro** hat **Apple** einen High-End-Ansatz geschaffen, bei dem Mixed Reality im Vordergrund steht. Trotz des kleiner ausfallenden Sichtfeldes als bei Konkurrenzprodukten bietet sie hochauflösende OLED-Displays mit hervorragender Bildschärfe bis zu den Rändern durch Speziallinsen. Des Weiteren besitzt sie einen Drehregler der stufenloses schalten zwischen VR und Passthrough-AR ermöglicht. Die Steuerung erfolgt ausschließlich durch präzises Hand- und Augen Tracking, welche aber bei schnellen Bewegungen Einschränkungen aufweist. Aufgrund des hohen Preises ist das Headset nicht für den breiten Markt geeignet. Sie bietet aber Unternehmen und Entwicklern viel Potential im Bereich Visualisierung und Design (Erl & (Erl & Danneberg, 2025).

Die **PlayStation VR2** ist speziell für die PlayStation 5 konzipiert und fokussiert sich stark auf Gaming. Mit hochauflösenden OLED-Displays, präzisem Tracking und haptischem Feedback in den Controllern und der Stirnhalterung bietet sie ein besonders immersives Spielerlebnis. Durch einen Adapter, welcher zusätzlich erworben werden muss, kann diese Brille auch an

einem PC betrieben werden. Allerdings muss dabei ein Verzicht von Funktionen wie haptisches Feedback, Eye-Tracking und 3D-Audio gemacht werden (Erl & Danneberg, 2025).

Die **Pico 4** hat technisch eine höhere Auflösung als ihr Konkurrenzmodell Quest 2, praktisch ist dieser Unterschied allerdings kaum sichtbar. Die Bildklarheit ist bis zu den Rändern hin sehr gut. Schnelle Kopfbewegungen sorgen allerdings für doppeltes Anzeigen von Bildinhalten und einem verschmierten Bildeindruck. Die Farben wirken dunkel und verwaschen. Eine experimentelle Version von Handtracking ist vorhanden aber nicht überzeugend, während das Tracking der Controller aber stabil und zuverlässig arbeitet. Durch den niedrigen Preis könnte sich Pico als Konkurrenz zu Meta etablieren (Erl & Danneberg, 2025).

Die **Pimax Crystal** hebt sich durch extrem hohe Auflösung und ein besonders breites Sichtfeld hervor. Sie richtet sich vor allem an Enthusiasten, die Wert auf maximale Bildqualität legen, beispielsweise in Racing- oder Flugsimulatoren. Suboptimal ist ihr hohes Gewicht, welches sich nach einiger Zeit im Nacken bemerkbar macht. Gleichzeitig ist sie teurer und technisch anspruchsvoller in der Einrichtung. Da es sich bei dieser VR-Brille um ein „Work in Progress“ Produkt handelt, stehen finale Tests und Berichte noch aus (Erl & Danneberg, 2025).

Neben den Consumer-Brillen existiert ein professioneller B2B-Markt. Geräte von **Varjo** oder die **HTC Vive Focus 3** sind speziell auf Unternehmensanwendungen ausgelegt. Sie bieten extrem hohe Auflösungen, präzises Tracking und erweitertes Mixed-Reality-Feature-Set. Typische Einsatzbereiche sind Industrie, Forschung, Training, Design und Simulationen. Hier stehen Genauigkeit, Zuverlässigkeit und die Integration in bestehende Arbeitsprozesse im Vordergrund, weniger Preis oder Unterhaltungswert. B2B-Brillen ermöglichen dadurch realitätsnahe Darstellungen und komplexe Trainings- oder Designprozesse, die im Consumer-Bereich nicht notwendig sind (Erl & Danneberg, 2025).

Für den Massenmarkt sind also vor allem die Meta Quest 3, beziehungsweise 3s, die Apple Vision Pro, PlayStation VR2 und Pico 4 geeignet. Sie kombinieren Mixed-Reality-Funktionen, gutes Tracking und attraktive Features für Gaming und Entertainment. Die Quest 3 bietet zurzeit das beste Preis-Leistungs-Verhältnis für den Konsumer.

Die Entwicklung und Verbesserung neuer Brillen werden kontinuierlich fortgeführt. Ab Sommer 2025 plant Samsung, ein eigenes XR-Headset auf den Markt zu bringen. Dieses entsteht in Zusammenarbeit mit Qualcomm und Google und ist für Unterhaltung sowie professionelle Anwendungen konzipiert. Die Veröffentlichung der Apple Vision Pro 2 wird frühestens Ende 2026 erwartet. Laut Gerüchten sollen zwei Versionen erscheinen, die eine höhere Leistung, eine verbesserte Darstellung und erweiterte KI-Anwendungen bieten. Valve arbeitet parallel an der Valve Index 2, die besonders für Gaming-Enthusiasten optimiert wird. Sie soll eine höhere Auflösung, ein größeres Sichtfeld, eine schnellere Bildwiederholrate und eine ergonomischere Gestaltung erhalten. Auch Asus plant ein neues Headset. Die ROG VR soll mit hochauflösender Grafik, präzisem Positionstracking und reaktionsschnellen Bildwiederholraten speziell für immersives Gaming ausgelegt werden. Lenovo entwickelt ein Headset, das auf dem Horizon-Betriebssystem läuft und sich gezielt an professionelle

Anwender im Unternehmensumfeld richtet. Für Ende 2026 wird die Veröffentlichung der Meta Quest 4 erwartet. Sie soll die soziale Interaktion durch Gesichtserkennung und Eyetracking verbessern, intensivere Mixed-Reality-Erlebnisse ermöglichen, eine vollständig integrierte KI enthalten und gleichzeitig komfortabler zu tragen sein (Metamandrill, 2025).

Insgesamt lässt sich ein klarer Trend erkennen. Bis ins Jahr 2025 wurde die Hardware deutlich leichtgewichtiger, benutzerfreundlicher und Akkulaufzeiten und Tragegefühl verbesserten sich zunehmend (Metamandrill, 2025; Molchanov, 2025). Brillen, die neu auf den Markt kommen, bieten hochauflösende Displays, leistungsfähigere Chips und neben der Bedienung mit Controllern auch Augen- und Handtracking (Molchanov, 2025). Technische Fortschritte bestehen auch in höheren Bildwiederholraten und hochauflösender Grafik. Weitere Neuerungen sind die Einbindung von KI und die Entwicklung von MR-Brillen (Metamandrill, 2025). Diese gewinnen an Bedeutung, da sie einen nahtlosen Übergang zwischen realer und virtueller Welt ermöglichen (Metamandrill, 2025; Molchanov, 2025). Trotz sinkender Preise sind aber gerade hochwertige Produkte immer noch zu kostenintensiv für einen Massenmarkt (Molchanov, 2025).

2.3 Virtuelle Tierbeobachtung

2.3.1 Tierbeobachtung über Zoos und digitale Bildungsangebote

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

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

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

Abbildung 1 - Augmentierter Wolf (WWF, 2024; eigene Aufnahme)

Abbildung 2 - Fehlerhafte Interaktion (WWF, 2024; eigene Aufnahme)

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

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

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

2.3.2 Verfügbare XR-Spiele

Um die Einsatzmöglichkeiten von XR für Tierbeobachtung und -fotografie besser einschätzen zu können, werden im Folgenden einige aktuell für Meta verfügbare Spiele betrachtet. Der Fokus liegt auf Anwendungen, die VR- oder MR-Technologien nutzen und Natur- und Tierbeobachtung, Fotografie oder Interaktion mit der Umgebung ermöglichen. Ziel ist es, zu untersuchen, welche Funktionen die Spiele bereits beinhalten, welche Ansätze gut umgesetzt sind und welche Aspekte bisher fehlen. Die gewonnenen Erkenntnisse dienen als Grundlage, um Potenziale für eine neue Anwendung abzuleiten und so zu entwickeln, dass realistische Tierbeobachtung, Interaktion und Wissensvermittlung sinnvoll miteinander kombiniert werden.

Aktuell zeigt sich, dass es bereits einige Virtual-Reality-Anwendungen im Bereich Natur- und Tiererleben gibt:

National Geographic Explore VR ermöglicht Expeditionen in die Antarktis und zum Machu Picchu. Der Schwerpunkt liegt auf Lernen durch selbst Erleben. Tiere wie Pinguine können in ihrer natürlichen Umgebung beobachtet werden, während die Landschaft erkundet wird. Die Spielenden können Fotos machen, was ein interaktives Element schafft und die Erlebnisse dokumentiert. Gleichzeitig vermittelt die Anwendung Wissen über Tierarten, Lebensräume und Umweltbedingungen. Allerdings ist die Interaktion auf Beobachten und Fotografieren begrenzt, und es fehlt ein strukturiertes System zur Wissensvermittlung oder Sammlung von Beobachtungen über mehrere Sessions hinweg (Bezmalinovic, 2022; Meta, 2025e).

BRINK Traveler ist eine VR-Anwendung, die virtuelle Reisen an reale Orte ermöglicht. Die Umgebungen wurden mithilfe von Photogrammetrie erstellt und wirken dadurch besonders realistisch. Die Nutzer können sich frei im Raum bewegen und die Umgebung aus verschiedenen Perspektiven erkunden. Ergänzend stehen interaktive Elemente wie ein Reiseführer mit Informationen zu den besuchten Orten zur Verfügung (*BRINK XR*, o. J.; Meta, 2025b)**.**. Der Schwerpunkt liegt also auf dem Erleben und Erkunden der Landschaften

Nature Treks VR legt den Fokus auf Achtsamkeit und Entspannung. Die Spielenden bewegen sich durch ruhige, meditative Landschaften, in denen verschiedene Tiere vorkommen. Die Anwendung erzeugt eine beruhigende Atmosphäre und kann zur Stressreduktion beitragen (Meta, 2025f). Ein Bildungsaspekt oder spielerische Interaktion ist jedoch nur minimal vorhanden, sodass die Verbindung von Naturerlebnis und Lernen nur bedingt realisiert wird.

Mit **Ocean Rift** kann eine virtuelle Unterwasserwelt erkundet werden. Die Umgebung ist frei begehbar, und Tiere wie Haie, Delfine oder Meeresschildkröten werden in 3D dargestellt. Informationen zu den Tieren werden eingeblendet, wodurch ein Lernaspekt entsteht. Dennoch bleibt die Interaktivität beschränkt. Beobachtung und freies Erkunden sind möglich, doch ein integriertes System zur Dokumentation von Beobachtungen oder Gamification-Elemente fehlen, sodass das Engagement über längere Zeit begrenzt sein kann (Meta, 2025g; Picselica, o. J.).

WildXR nutzt 360°-Videos, um Tiere in ihren Lebensräumen zu zeigen. Der Fokus liegt auf Artenschutz, Verhaltensweisen und der Vermittlung von Informationen über Ökosysteme. Die Anwendung ist sehr realitätsnah und informativ, bietet aber keine aktive Interaktion oder spielerische Elemente. Nutzer können die Umgebung nur passiv betrachten. Die fehlende Einbindung von Gamification oder Mixed-Reality-Elementen reduziert die Möglichkeiten, das Lernen langfristig motivierend zu gestalten (Meta, 2025h).

Im Bereich Augmented- und Mixed-Reality sind die Angebote deutlich eingeschränkt:

Living Room ermöglicht, Tiere direkt in der eigenen Umgebung zu betrachten. Entweder als Miniaturen auf dem Tisch oder lebensgroß im Raum. Die Darstellung ist cartoonhaft und vermittelt keinen realistischen Eindruck der Tiere. Zwar erlaubt die Anwendung ein räumliches Platzieren und Beobachten, jedoch fehlt ein pädagogischer Fokus. Es gibt keine strukturierte Wissensvermittlung, keine Aufgaben oder Belohnungen, und die Interaktion ist auf die reine Platzierung und Beobachtung beschränkt (Meta, 2025d; Streule, o. J.).

Animalz VR erlaubt die Beobachtung realistisch dargestellter Tiere in der eigenen Umgebung. Die Tiere können frei positioniert werden und bewegen sich in der virtuellen Umgebung. Dennoch fehlt ein edukativer oder spielerischer Ansatz. Die Anwendung bietet keine Möglichkeit, Beobachtungen zu dokumentieren, Lernfortschritte zu verfolgen oder motivierende Aufgaben zu erfüllen. Auch die Interaktion mit den Tieren ist sehr begrenzt, sodass der langfristige Lern- und Erlebniswert eingeschränkt bleibt (Meta, 2025a).

Die Anwendung **ZOSU Zoo** bietet eine virtuelle Zoo-Umgebung mit über 140 verschiedenen Tieren und 60 Dschungelpflanzen. Neben einem klassischen Zoo-Szenario („Zoo Land“) steht mit ZOSU Ocean eine Erweiterung mit 30 Ozean-Leveln zur Verfügung. Durch die Integration von Mixed-Reality- und Handtracking-Funktionen können die Nutzer einen eigenen Zoo in der realen Umgebung gestalten. Seit der Erstveröffentlichung im Mai 2022 wurde die Anwendung kontinuierlich erweitert. Im Juli 2024 kamen die Mixed-Reality- und Handtracking-Features hinzu, im Oktober 2024 neue Tierarten wie Buckelwal, Schildkröte und Dinosaurier. Im Februar 2025 folgte schließlich ein 360°-Strandvideo mit interaktiven Mini-Tieren. (Meta, 2025i)**.**

Die Analyse der aktuellen VR-, AR- und MR-Anwendungen zeigt, dass VR-Anwendungen in Bereich Natur- und Tiererfahrungen derzeit am weitesten entwickelt sind. AR- und MR-Anwendungen gibt es allerdings noch wenige. Bei den vorhandenen werden Tiere entweder cartoonhaft dargestellt oder es fehlen Interaktivität bzw. edukativer Mehrwert. Mixed-Reality-Anwendungen, die realistische Darstellung, aktive Beteiligung wie z.B. Fotografie, spielerische Elemente und einen edukativen Mehrwert kombinieren, wurden bei der Recherche nicht gefunden.

Hieraus ergibt sich eine deutliche Chance für die geplante Anwendung. Eine Mixed-Reality-Anwendung kann die Vorteile beider Welten verbinden, indem virtuelle Tiere realitätsnah in die eigene Umgebung integriert, aktive Beobachtung und Fotografie ermöglicht und die Erlebnisse in einem digitalen Entdeckertagebuch dokumentiert werden. Durch spielerische Elemente lässt sich das Engagement langfristig steigern, während Lernen und Naturerlebnis miteinander verknüpft werden. Auf diese Weise kann Bildung, Unterhaltung und Nachhaltigkeit in einer Plattform vereint werden.

3 Anforderungen

Tabelle 1 gibt einen detaillierten Überblick über alle Anforderungen an die MR-Anwendung. Jedes Kriterium ist dabei einer Priorität Muss, Soll oder Kann zugeordnet. Anhand der Priorisierung lässt sich sofort erkennen, welche Anforderungen für die Umsetzung besonders wichtig sind. Die Tabelle deckt verschiedene Bereiche des Spiels ab, unter anderem die Gestaltung der Spielumgebung und des Tagebuchs, das Verhalten der Tiere und weitere relevante Funktionen und Inhalte. Sie bietet somit eine strukturierte Grundlage für die Planung und Entwicklung des Spiels.

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
1.4	Umgebung	Die Anwendung bietet unterschiedliche Tageszeiten und Lichtstimmungen (z. B. Mittag, Dämmerung, Nacht)	Kann
2.1	Tiere	Tiere verhalten sich realistisch (z. B. flüssige Bewegungen, 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
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	Kann
5.	Plattform	Die App ist für die Meta Quest 3 konzipiert	Muss
6.1	Benutzeroberfläche	Steuerung der Kamera und Navigation durch das Tagebuch sind intuitiv	Sollte
6.2	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	Umweltelemente (z. B. Bäume, Felsen) erscheinen in der realen Umgebung	Sollte
8.3	Mixed Reality	Nutzer können mit der Umgebung interagieren	Kann
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	Kann

Der Fokus der Anwendung liegt auf der Umsetzung der Anforderungen, die ein immersives Mixed-Reality-Erlebnis ermöglichen. Besonders wichtig ist eine frei begehbare Umgebung, das realistische Verhalten der Tiere sowie die Integration der virtuellen Tiere in die reale Umgebung. Zentrale Funktionen wie Fotografie und das Entdeckertagebuch sind als Muss-Anforderungen definiert, da sie die Hauptfunktionen der Anwendung darstellen. Wichtige Anforderungen, die das Spiel erweitern, aber nicht unbedingt für den Prototypen notwendig sind, werden als Sollte klassifiziert. Dazu zählen zum Beispiel dynamische Tageszeiten, daran angepasste Auftrittswahrscheinlichkeiten der Tiere oder eingeblendete virtuelle Umgebungsobjekte wie Bäume oder Felsen. Optionale Details wie Belohnungssysteme, Multiplayer oder erweiterbare Umgebungen werden als Kann-Anforderungen markiert, da sie zusätzlichen Nutzen bieten, aber nicht zwingend erforderlich sind. Die Anwendung ist so konzipiert, dass sie flexibel erweitert werden kann, zum Beispiel durch neue Umgebungen, Tiere oder Kamerafunktionen wie Zoom oder Blitz. Gleichzeitig wird Wert auf eine intuitive Benutzeroberfläche gelegt, um eine einfache Bedienung für unterschiedliche Nutzergruppen zu ermöglichen.

Durch die Priorisierung lässt sich deutlich erkennen, welche Funktionen für den grundlegenden Prototypen entscheidend sind und welche als Ergänzung hinzugefügt werden können, um Motivation, Lernfortschritt und Spaß zu steigern. Dabei wird gleichzeitig Wert auf intuitive Bedienbarkeit, Erweiterbarkeit und die Vermittlung von Wissen über Tiere gelegt, sodass die Anwendung sowohl Bildung als auch Naturerlebnis in Einem bieten kann.

4 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.

4.1 Hardware

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

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

4.2 Entwicklungsumgebung

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

Unity Unity bietet eine Entwicklung mit C# und wird als einsteiger- und benutzerfreundlich beschrieben (Dadson, 2023; Frank, 2022; Rocketbrush, 2024). Schnelles Prototyping soll mit Unity möglich sein und die Einarbeitung wird durch eine große Community und eine gute Dokumentation unterstützt (Dadson, 2023). Zudem bietet Unity einen kostenlosen Unity

[3] Hochauflösend ist an dieser Stelle ein Werbeversprechen und muss nicht mit der subjektiven Wahrnehmung zusammenfallen.

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

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

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

4.3 Programmiersprache

Die Wahl der Programmiersprache hängt stark mit der gewählten Umgebung sowie den technischen Anforderungen der Zielplattform zusammen. In diesem Projekt wurde Unity als Umgebung in Kombination mit der Meta Quest 3 als Zielgerät gewählt. Dennoch ist es sinnvoll,

[4] Spiele mit hohem Budget und meist hochwertiger Grafik und aufwendigen Spielmechanismen. Meist von großen Entwicklerstudios produziert.

auch die Programmiersprachen anderer gängiger Umgebungen zu betrachten, um die Entscheidung besser einordnen zu können.

C# ist wie bereits erwähnt die zentrale Programmiersprache von Unity. Es ist eine universell einsetzbare, objektorientierte Programmiersprache, die sich gut für verschiedene Softwareprojekte eignet. Sie ist leicht zu erlernen, bietet eine automatische Speicherverwaltung und eine klare, einfache Syntax, was besonders für Einsteiger in die Spiele- und AR/VR-Entwicklung vorteilhaft ist. Darüber hinaus ist C# stark in die Unity-Engine integriert. Unity nutzt C# zur Skripterstellung, was eine schnelle Umsetzung von Ideen, Prototypen und interaktiven Anwendungen ermöglicht. Zudem ist C# durch die umfangreiche Unity-Community und zahlreiche Tutorials stark unterstützt, was die Einarbeitung erleichtert (Circuit Stream, 2022).

Im Gegensatz dazu ist **C++** eine leistungsfähige Programmiersprache, die hauptsächlich in der Unreal Engine verwendet wird. Sie erlaubt manuelle Speicherverwaltung und Kontrolle über Systemressourcen. Dadurch lassen sich Projekte mit hohen Anforderungen an Leistung und Effizienz optimal umsetzen. C++ erfordert jedoch eine längere Einarbeitungszeit und tiefere Programmierkenntnisse. Für Entwickler, die präzise Kontrolle über Hardware und Software benötigen, bietet C++ klare Vorteile, während Anfänger oft von der Komplexität und höheren Fehleranfälligkeit abgeschreckt werden (Circuit Stream, 2022).

Durch das **Blueprint-System** der Unreal Engine können Entwickler die gesamte Anwendungslogik über Drag and Drop erstellen, ohne selbst Code schreiben zu müssen. Dies erleichtert die schnelle Umsetzung von ersten Ideen und Prototypen, insbesondere für Designer oder Einsteiger, die noch nicht mit C++ vertraut sind (Circuit Stream, 2022).

Die Wahl zwischen C# und C++ sollte immer in Abhängigkeit von der Projektgröße, den Zielen und der Zielplattform erfolgen. Für dieses Projekt stellt die Kombination aus Unity und C# eine geeignete Wahl dar, da sie eine schnelle Entwicklung, einfache Handhabung und starke Community-Unterstützung bietet. C++ in Unreal Engine bietet hingegen Vorteile bei Leistung und Kontrolle, ist jedoch komplexer in der Umsetzung. Somit ist C# in Unity für die Meta Quest 3 die optimale Lösung, um effizient Anwendungen zu entwickeln, während C++ eher für größere und leistungsorientiertere Projekte geeignet ist.

4.4 Architekturmodell

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

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

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

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

Jedoch existieren auf dem MVC-Prinzip weiter entwickelte Architekturmodelle, die an XR-spezifische Bedarfe angepasst sind. So entwickelten (Benbelkacem et al., 2019) mit “MVC-3D” eine Erweiterung, um XR-Anforderungen zu adressieren. Kernaspekte waren das Hinzufügen einer “Library”-Komponente und das Weiterführen der View zur “interactive View”. Die Library kapselt komplexe Algorithmen, wie sie für Tracking, Gestenerkennung oder

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

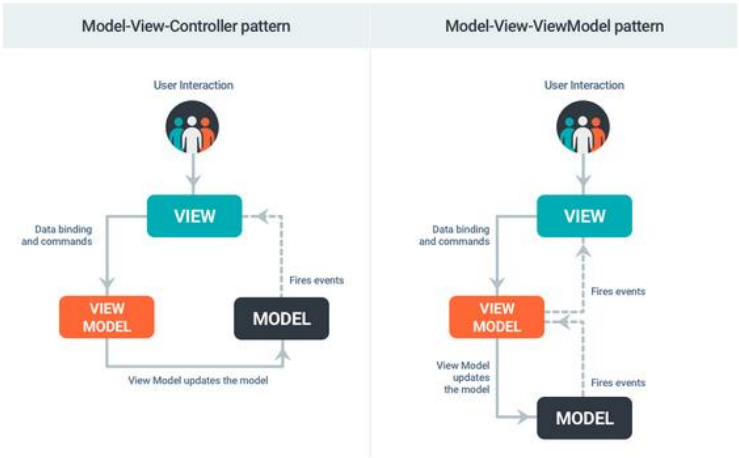

Abbildung 3 - Unterscheidung MVC und MVVM (MVVM Pattern on Android | WOXAPP, o. J.)

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

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

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

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

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

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

MVVM wird aufgrund der erhöhten Komplexität für einen zeitgleich geringen Umfangs des Backends verworfen.

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

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

[5] Beispiele für 2D und 3D-Anwendungen finden sich in (Heckl et al., o. J.; Hosseini, 2025; Unity Technologies, 2024; Yang, 2017/2024).

werden und wird als Basisarchitekturmodell für das Frontend festgelegt. Es unterstützt zudem eine plattformunabhängigere Entwicklung und adressiert damit eine zentrale Herausforderung aktueller XR-Anwendungen (Molchanov, 2025).

Damit bei der Entwicklung auch spezielle Anforderungen für XR-Anwendungen adressiert werden können, wird dieses Architekturmodell auf Basis von Benbelkacem et al., 2019 und Benbelkacem et al., 2020 erweitert. Um Tracking und komplexere Berechnungen zu kapseln, wird dem klassischen MVVM eine „Library"-Schicht hinzugefügt. Weiter wird die Trennung in Frontend und Backend als Voraussetzung für die Möglichkeit von kollaborativen Systemen verstanden. Damit Interaktionen zwischen realen und virtuellen Elementen möglich werden, soll die View Komponenten beinhalten, die auf unterschiedliche Nutzereingaben wie beispielsweise Gesten oder auch Controller reagieren können. Eine ausführliche Erläuterung des weiterentwickelten Architekturmodells findet sich in Kapitel 0 Das Backend wird mit der MVC-Struktur implementiert. Um den Zugriff auf die Datenbank zu kapseln, wird jedoch das „Repository" als weitere Schicht eingefügt. Das Repository kapselt den Zugriff auf die Datenbank beziehungsweise auf die lokale Datenbanksimulation. Ziel ist es, damit ein klar strukturiertes und modulares Backend zu implementieren, das so nicht nur erweiterbar und einfach wartbar ist, sondern zusätzlich noch Skalierbarkeit ermöglicht. Das MVC-Muster trennt dabei die Verantwortlichkeiten in Datenstruktur, Logik und Interaktionsschnittstelle. Im Folgenden werden die einzelnen Schichten genauer beschrieben.

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

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

View Die View wird im Backend nicht als graphische Benutzeroberfläche implementiert, sondern liegt in Form von API-Endpunkten vor. Diese Endpunkte stellen die sichtbare Oberfläche des Backends dar, indem sie den Zugriff auf ausgewählte Funktionen und Daten

ermöglichen und als Kommunikationsschnittstelle genutzt werden. So kann beispielsweise das Frontend auf die Daten einzelner Tiere zugreifen oder über einen Endpunkt einen neuen Account erstellen. Die technische Umsetzung erfolgt über standardisierte REST-Schnittstellen, was die Interoperabilität mit anderen Systemen und Plattformen sicherstellt.

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

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

Frontend. Kernaspekte der Weiterentwicklung sind also eine zusätzliche Schicht, zusätzliche Komponenten in der View und die Aufteilung in Frontend und Backend als Teil des Architekturmodells.

5 Konzept

5.1 Aufbau der Anwendung

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

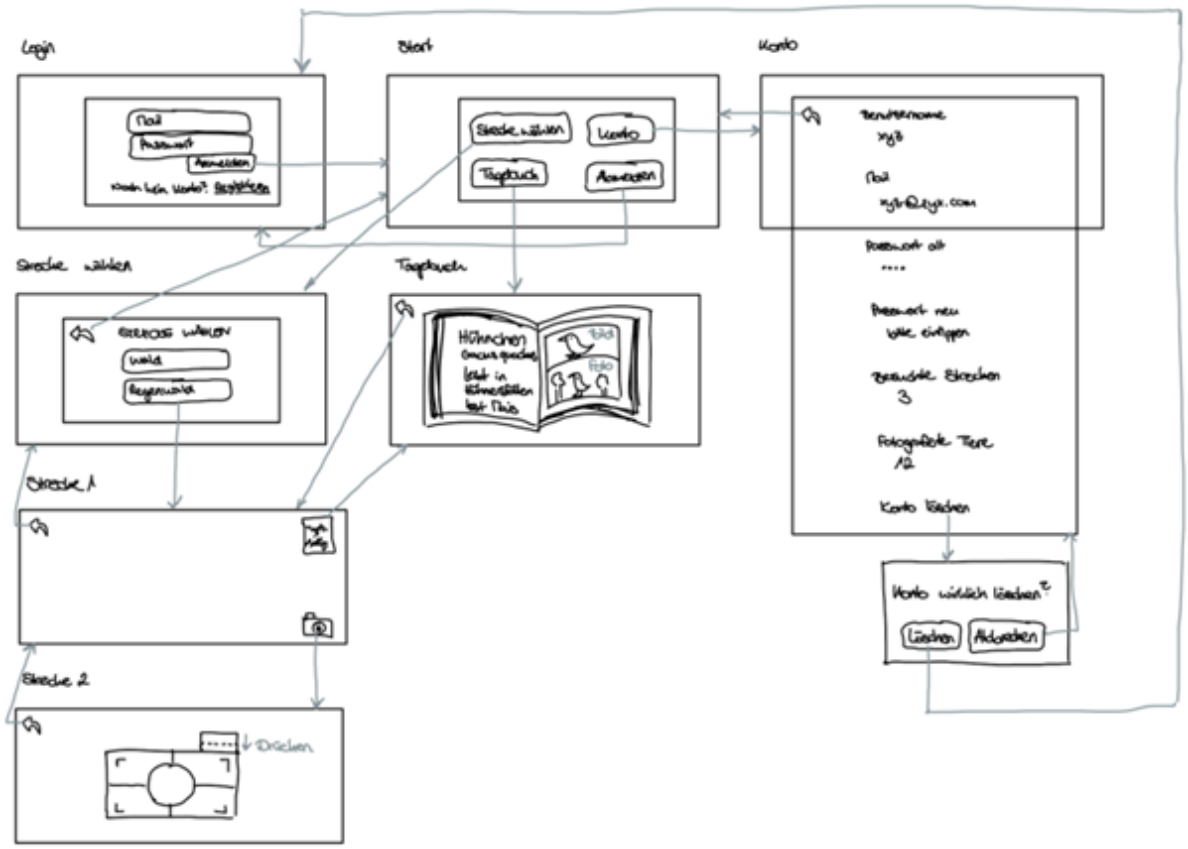

Abbildung 4 - Mockup der Anwendung (eigene Zeichnung)

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

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

Die Option Strecke wählen öffnet eine Szene, in der verschiedene Umgebungen – etwa ein Wald, die Antarktis oder der Regenwald – auswählbar sind. Jede Strecke bietet eigene Tiere und Umgebungselemente. Nach Auswahl einer Strecke beginnt das Spiel (Szene „Strecke 1“), in dem Tiere einzeln oder zeitgleich zufällig erscheinen können. Ihr Verhalten orientiert sich

am realen Vorbild: Eine Schnecke kriecht unbeirrt über den Weg, während ein Reh bei Unterschreiten einer bestimmten Distanz flieht. Im MR-Kontext stellt die Interaktion zwischen virtuellen Elementen und realer Umwelt eine besondere Herausforderung dar: Beispielsweise darf ein Reh nicht durch einen realen Baum laufen oder in der Luft schweben, sondern muss korrekt verdeckt werden und auf dem realen Untergrund stehen. In Szene „Strecke 1“ können Nutzer außerdem ins Tagebuch oder zur Kamera wechseln.

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

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

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

5.2 Grundlegende Architektur

Im Folgenden wird die Architektur des Prototyps konzeptionell erläutert. Ziel ist es, prototypisch eine MR-Anwendung zu entwickeln, die es ermöglicht, über die zur Verfügung gestellte Meta Quest 3 animierte Tiere im realen Raum zu sehen und zu fotografieren. Darüber hinaus soll eine Interaktion zwischen dem animierten Tier und der realen Umgebung ermöglicht werden, zum Beispiel indem es bei einer zu geringen Distanz flieht und verschwindet. Bei der Entwicklung sollen mit der Architektur die Faktoren Skalierbarkeit und Multiplayermodus schon berücksichtigt werden, auch wenn diese nicht direkt implementiert werden. Der Faktor Skalierbarkeit soll über lose gekoppelte Schichten erreicht werden, sodass einfach Erweiterungen an Programm und Programmcode vorgenommen werden können. Zudem können mit diesem Faktor einzelne Schichten auf unterschiedlichen Systemen laufen und so kann bei Bedarf Rechenleistung hinzugezogen werden. Damit eine Einarbeitung schnell erfolgen kann und Erweiterungen einfach implementiert werden können, wird ein strukturierter und leicht verständlicher Aufbau vorausgesetzt. Um einen Multiplayermodus zu ermöglichen, wird eine zentrale Datenbasis (Server), eine Synchronisation zwischen darauf zugreifenden Parteien (Clients) und eine konfliktfreie Zustandsverwaltung benötigt. Aus diesen Gründen soll der Prototyp in drei Schichten unterteilt werden. Wie bereits angemerkt sollen die beiden Schichten Frontend und Backend unter Verwendung geeigneter Architekturmuster entwickelt werden.

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

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

Die letzte Schicht ist das Frontend. Es ist dafür zuständig, die bereitgestellten Daten (graphisch) darzustellen. Diese Schicht beinhaltet die Benutzeroberfläche[6] und somit die Schnittstelle zwischen Anwendung und Mensch. Da auch die Darstellung von Inhalten einer gewissen Logik

[6] Die 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.

bedarf, enthält auch das Frontend Logiken. Diese beziehen sich aber auf die Präsentation von UI-Elementen und dürfen nicht mit der Backend-Logik verwechselt werden.

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

5.2.1 Datenbank

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

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

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

5.2.2 Backend

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

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

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

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

Controller Die zentrale Stelle im Backend des Prototyps ist der Controller. Der stellt nicht nur die Kommunikation mit der Außenwelt her, sondern kommuniziert auch mit dem Model und dem Repository und stößt somit die Kommunikation mit der Datenbasis an. So fungiert der

Controller als Koordinationsinstanz, die Datenfluss, Verarbeitungslogik und Sicherheit kontrolliert. Zusätzlich enthält der Controller weitere Logik. Sollte eine Person einen Account anlegen, wird so beispielsweise durch den Controller auch das Erstellen eines zugehörigen Tagebuchs angestoßen. Im Rahmen einer möglichen Erweiterung zum Multiplayersystem wäre der Controller zudem die zentrale Stelle zur Synchronisierung von Zuständen und zur Validierung eingehender Änderungen durch verschiedene Clients und zur Konfliktvermeidung.

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

5.2.3 Frontend

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

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

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

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

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

ViewModel Das ViewModel ist das zentrale Steuerungselement in der Architektur. Zentral für diese Funktion sind sogenannte Events[7], Observer-Patterns[8] und auch Reactive-Patterns[9]. Das ViewModel empfängt Änderungen von View oder Model, verarbeitet diese und leitet sie weiter. Dabei müssen View und Model nicht direkt miteinander kommunizieren. Beispielsweise empfängt das ViewModel bei der Änderung einer E-Mail-Adresse die neue Adresse und aktualisiert automatisch das Model, wodurch wiederum eine Modeländerung dem ViewModel gemeldet wird und infolgedessen automatisch die View und damit die Anzeige geändert wird. Dafür müssen die Verbindungen zwischen View und ViewModel zur Laufzeit überwacht werden. Das geschieht, indem sogenannte Properties im ViewModel mit UI-Elementen verknüpft werden. Properties können als Felder oder Variablen verstanden werden, die zusätzlich eine Logik enthalten. Diese Logik kann zum einen den Zugriff auf die Variable steuern, oder aber die Möglichkeit bieten, ein UI-Element zu verknüpfen. Diese Verknüpfung ermöglicht eine automatische Synchronisation von Eingabe und Variablenwert bei Änderungen (Stonis, 2024). So kann im vorigen Beispiel das Eingabefeld für die E-Mail-Adresse mit einer E-Mail-Property im ViewModel verknüpft werden. In diesem Ablauf sorgt dann das Event bei der Änderung der E-Mail-Property dafür, dass im Rahmen des Observer-Patterns alle relevanten Elemente informiert werden, und das Reactive-Pattern übernimmt die direkte Synchronisation zwischen Property, Model und View. Ein weiterer Vorteil des

[7] Event: Änderungssignal

[8] Observer-Pattern: Objekt „lauscht“ auf Änderungen eines anderen Objekts, erfährt Änderungen sofort

[9] Reactive-Pattern: Systematische Beobachtung von Datenflüssen und Reaktion darauf

Architekturmodells, neben diesem Verhalten und der strikten Trennung von Aufgabenbereichen, besteht in der Möglichkeit, Berechnungen aus der Library hinzuzuziehen. Ein Beispiel hierfür ist die Flucht eines Tieres, bei der dem ViewModel eine Nutzeraktion in Form einer zu geringen Distanz zwischen Person und Tier von der View gemeldet werden kann. Das ViewModel setzt dann die Logik „Flucht“ um und nutzt dabei die Fluchtimplementierung des jeweiligen Tieres aus der Library-Komponente. Ist der Bewegungsablauf abgeschlossen, kann das ViewModel dem Model die neue Position des Tieres weiterleiten, die daraufhin an das Backend weitergeleitet und dort in einer Datenbank gespeichert wird. Diese Änderung an der Datenbasis, und damit am Model, wird zurück an das ViewModel geleitet. Dort wird eine Aktualisierung der Anzeige, also View, angestoßen. Umsetzbar ist dieses Verhalten mittels Data Binding, welches UI-Elemente automatisch mit den entsprechenden Properties verknüpft. Wie bereits erläutert, kann sich bei Änderung an der Datenbasis die Anzeige automatisch aktualisieren und umgekehrt bei einer Nutzereingabe automatisch die Datenbasis ändern.[10]

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

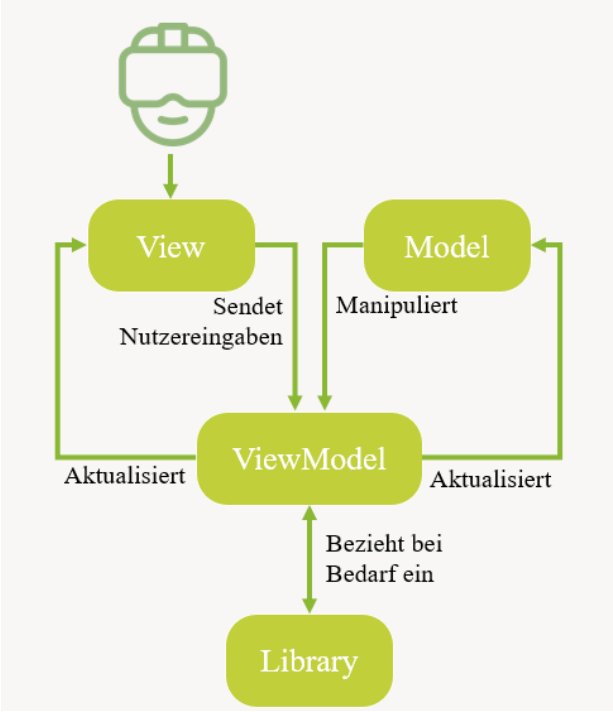

Abbildung 5 - MVVM-3D (eigene Darstellung)

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

[10] Hierbei soll nur von den technischen Möglichkeiten gesprochen werden. Sicherheitskonzepte sollten in einer realen Anwendung die direkte und ungeprüfte Manipulation an der Datenbasis verhindern.

Wie im Kapitel 4.4 Architekturmodell schon angemerkt, gibt es für die Entwicklung von kollaborativen Systemen drei Methoden. Mit der Grundstruktur von Datenbank, Backend und Frontend ist allerdings die Methode „Centralized Mode“ am meisten geeignet. Diese Methode beschreibt einen zentralen Server als „Single Source of Truth“ und mehrere Clients, die jeweils auf denselben Datenstand zugreifen (Benbelkacem et al., 2020; Dewan, o. J.; Fleury, o. J.). Das Backend und die Datenbank stellen im Kontext des Prototyps den Server dar, der alle Anfragen und Datenoperationen verarbeitet. Mehrere Instanzen des Frontends übernehmen die Rolle von unterschiedlichen Clients und können unabhängig voneinander auf das Backend/den Server zugreifen. Dieses Modell ermöglicht es mehreren Benutzern, gleichzeitig mit derselben Anwendung zu interagieren, wobei Konsistenz und Synchronisation durch den zentralen Server gewährleistet werden. Für eine Umsetzung würde das bedeuten, dass zum Beispiel Koordinaten von Tieren oder getrackte Gesten zwingend an das Backend gesendet werden und dort entweder in einem temporären oder persistenten Speicher gehalten werden müssen. Weiterhin wird ein Mechanismus benötigt, der Änderungen an die Clients propagiert, damit diese synchronisiert werden können. Die Kommunikation über zyklische GET-Anfragen ist hierfür nicht ausreichend, da sie nicht nur ineffizient, sondern auch anfällig für Verzögerungen und Synchronisationsprobleme ist. Dies setzt nicht nur eine Anfrage-basierte Kommunikation voraus (Pull-Prinzip), sondern auch eine Realisation von serverseitigen Benachrichtigungen der Clients (Push-Prinzip). Da das Backend sowie die relevanten Teile des Frontends (Services) in C# implementiert werden, ist eine C# beziehungsweise .NET-Lösung ausreichend. Eine Möglichkeit in diesem Kontext wäre der Einsatz von SignalR. Diese Bibliothek ermöglicht es, serverseitige Ereignisse unmittelbar an die verbundenen Clients weiterzuleiten, wodurch eine bidirektionale Kommunikation zwischen Server und Clients hergestellt wird. Dabei können Nutzeraktionen auf mehreren Clients in Echtzeit synchronisiert werden (Fletcher, 2023).

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

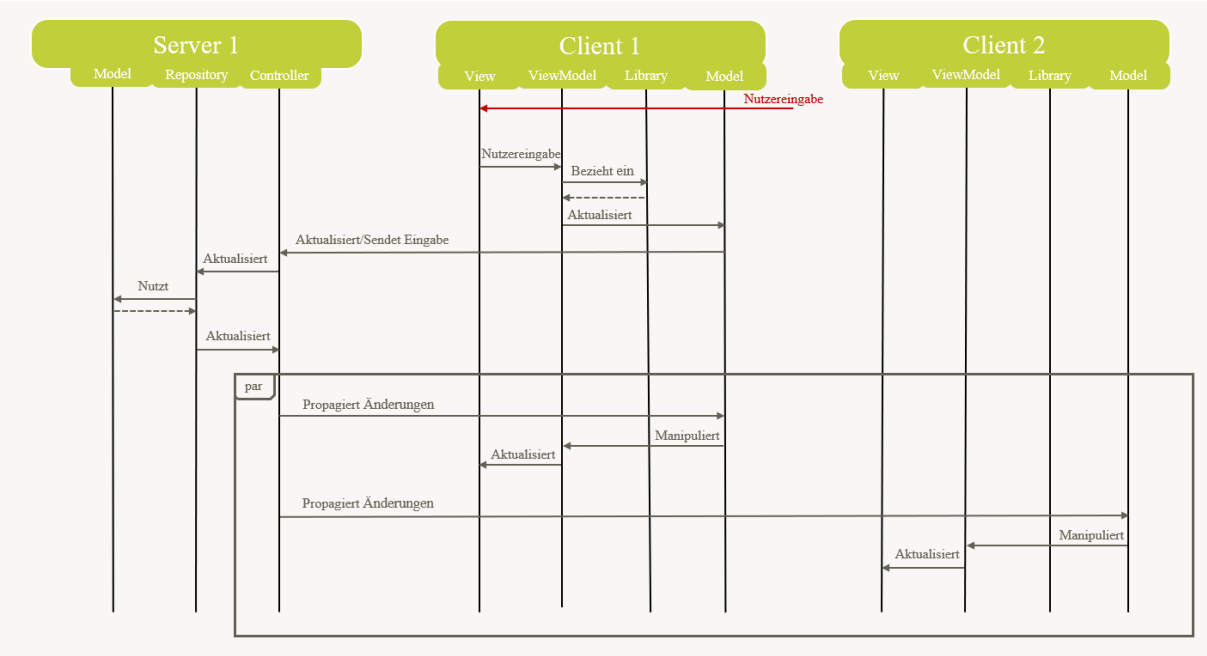

Abbildung 6 - Ablauf Kollaboration (eigene Darstellung)

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

5.3 Interaktion

Die Interaktion mit der MR-Anwendung erfolgt über die Meta Quest 3. Genutzt wird eine Kombination verschiedener Eingabemethoden, um ein möglichst immersives, intuitives und flexibles Nutzererlebnis zu schaffen. Steuerungselemente sind die Controller, die für Navigation, Menüauswahl, Auslösen von Aktionen und Interaktion mit den virtuellen Tieren verwendet werden. Über die Controller können Nutzer Objekte auswählen, Fotos aufnehmen, Bewegungen auslösen und bestimmte Interaktionsfunktionen starten. Parallel dazu ermöglicht das Handtracking natürliche Gesten wie Zeigen, Greifen, Winken oder Wischen, wodurch die Interaktion unmittelbar und realitätsnah gestaltet wird. Texteingaben dienen der Anmeldung, Profilerstellung und Personalisierung, z. B. für das digitale Entdeckertagebuch, in dem Beobachtungen, Fotos und Kommentare gespeichert werden.

Ein wichtiges Element stellt die vordefinierte, kollisionsfähige Umgebung dar. Diese wird über ein Mesh abgebildet. Ein Mesh ist ein 3D-Gitter, das aus vielen kleinen Polygonen (meist Dreiecke oder Vierecke) besteht. Diese Polygone setzen sich wie Puzzleteile zusammen und bilden die Oberfläche eines virtuellen Objekts. Ein Mesh simuliert damit den Boden, die Wände, Möbelstücke und andere Hindernisse (Awati, 2024). Das Mesh bildet also den Raum ab, indem sich virtuelle Objekte befinden und bewegen können. Die Tiere reagieren auf die Grenzen des Meshs. Durch die Kollisionserkennung können so Fluchtverhalten, Wegfindung und Interaktion mit realen Objekten realistisch umgesetzt werden können. So können auch Tiere hinter realen Gegenständen, die sich im gescannten Raum befinden, verschwinden. Wenn das

Mesh also den Raum nicht bis ins letzte Detail korrekt erfasst, können unrealistische Effekte wie Laufen durch reale Gegenstände auftreten.
Konzeptionell wäre für ein Szenario wie einen Waldspaziergang ein kontinuierliches Scannen der Umgebung erforderlich. Dabei müsste das Headset in einem bestimmten Radius um die Nutzerposition ständig neue Scans erstellen. Diese müssten in regelmäßigen Abständen mit den bestehenden Daten abgeglichen und zusammengeführt werden. Dieses Vorgehen ist technisch anspruchsvoll. Es besteht ein hoher Speicherbedarf und Rechenaufwand und es wäre gegebenenfalls eine Cloud-Auslagerung erforderlich, für die eine stabile Internetverbindung notwendig ist. Zudem weist Meta in seinen Sicherheits- und Garantiehinweisen ausdrücklich darauf hin, dass die Quest-Headsets nicht für die Benutzung draußen empfohlen sind (Meta, o. J.).
Für den Prototyp wird daher bewusst auf einen einzelnen, statisch gescannten Raum zurückgegriffen. Das Mesh wird dafür einmalig erstellt und dient danach unverändert als Grundlage für Interaktion und Kollisionserkennung. Diese kann so im Prototypen gut getestet und evaluiert werden. Diese Vorgehensweise erhöht die Stabilität und Performance des Prototyps, während die Integration dynamischer Scanverfahren zu einem späteren Zeitpunkt technisch möglich bleibt.

Die Umgebung ist frei begehbar, innerhalb der Grenzen des definierten Meshes. Nutzer können sich umsehen, bewegen und Tiere aus unterschiedlichen Perspektiven beobachten. Die Tiere reagieren auf die Bewegung des Nutzers. Beispielsweise fliehen sie, wenn der Nutzer ihnen zu nahekommt. Technisch wird dies dadurch umgesetzt, dass die Position der Quest 3 im Raum kontinuierlich erfasst und in der Anwendung bereitgestellt wird. Parallel dazu ist die Position jedes Tieres im Code verfügbar. Für jedes Tier wird ein individueller Fluchtradius definiert, der in der Datenbasis hinterlegt ist. Die Distanz zwischen Nutzer und Tier wird in jedem Frame ausgelagert in der Library berechnet. Liegt die Distanz unterhalb des festgelegten Fluchtradius, wechselt das Tier seinen internen Status von „normale Bewegung“ auf „Flucht“.
Eine einfache Variante besteht darin, das Tier direkt in entgegengesetzter Richtung zur Nutzerposition zu verschieben, indem ein normalisierter Richtungsvektor berechnet und mit einer Geschwindigkeit multipliziert wird. Der eigentliche Zustandswechsel von normaler Bewegung zu Flucht wird in der Library gekapselt, die kontinuierlich die Abstände prüft und den passenden State setzt. Damit bleibt die Logik im ViewModel schlank, während die Bibliothek die Distanzberechnung und die Zustandsverwaltung für die Animation der virtuellen Tiere übernimmt.

Eine wichtige Interaktionsfunktion ist außerdem die Fotografie der Tiere. Über einen Knopfdruck auf dem Controller kann ein Foto aufgenommen werden, das automatisch im digitalen Entdeckertagebuch gespeichert wird. Wird über den Controller ein Foto ausgelöst, so greift die Anwendung zunächst auf das aktuell aktive Tier zu. Da im Prototyp jeweils nur ein Tier gleichzeitig vorhanden ist, erfolgt die Abfrage eindeutig über die Referenz auf dieses Tierobjekt. Aus dessen Namen wird anschließend ein eindeutiger Dateiname gebildet, unter dem das aufgenommene Bild im vorgesehenen Verzeichnis für Tierfotos gespeichert wird.

Parallel dazu wird ein Tagebucheintrag erzeugt, der den Tiernamen, die Tier-ID, den Speicherpfad des Fotos und den Zeitpunkt der Aufnahme enthält. Die Verbindung zwischen Tagebuch und Tagebucheintrag (DiaryDiaryEntry) wird vom ViewModel angelegt und an den zuständigen Service weitergegeben. Der Service ruft den vorhandenen API-Endpunkt createLinkToDiary auf, wodurch die Verknüpfung hergestellt und der Eintrag dauerhaft im digitalen Entdeckertagebuch gespeichert wird. Nach erfolgreicher Speicherung ist der Eintrag unmittelbar in der Benutzeroberfläche sichtbar.

Für eine zukünftige Multiplayer-Erweiterung ist das System so vorbereitet, dass Nutzeraktionen in Echtzeit synchronisiert werden können. Aktionen wie das Zeigen auf ein Tier oder das Aufnehmen eines Fotos könnten über einen zentralen Server an andere Clients weitergegeben werden. Dadurch können mehrere Personen gleichzeitig dieselbe Szene erleben und interagieren, während Konsistenz und Synchronisation gewährleistet bleiben. Auch Gesten und Bewegungen könnten so übertragen werden, sodass die Nutzer gemeinsam ein realistisches Tierbeobachtungserlebnis teilen.

Zusammengefasst kombiniert das Interaktionskonzept mehrere Eingabemethoden, eine kollisionsfähige, frei begehbare Umgebung, realistisches Tierverhalten, Fotografie, digitale Dokumentation im Entdeckertagebuch und mögliche Multiplayer-Interaktionen. Durch die Fokussierung auf einen statischen Raum im Prototyp werden Performance und Stabilität gesichert, gleichzeitig bleibt das Konzept flexibel für zukünftige Erweiterungen, Raumscanning oder Multiplayer-Funktionen.

6 Prototypenentwicklung

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

6.1 Datenbank

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

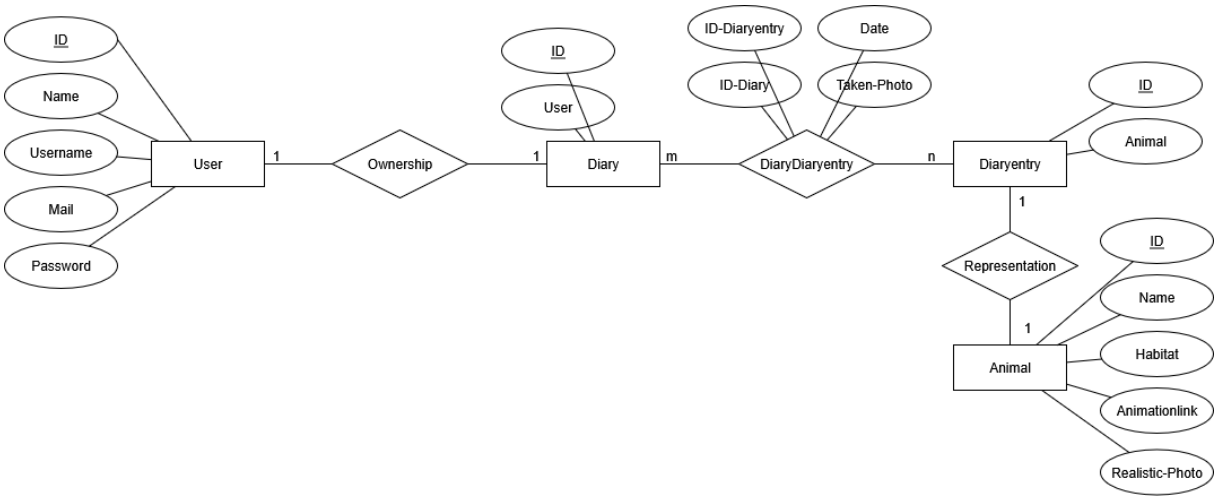

Abbildung 7 - ER-Diagramm Datenbank (eigene Darstellung)

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

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

[11] Es wird eine Unterscheidung zwischen einem realistischem Foto, also einem Foto eines realen Tieres und einem aufgenommenem Foto, also dem Foto, das ein User in der Anwendung von einem virtuellen Tier aufnehmen kann, gemacht.

Die Authentifizierung kann dann darüber abgewickelt und sensible Daten können dort sicher gespeichert werden.

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

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

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

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

6.2 Backend

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

Alle Repositories sind gleich aufgebaut. Sie implementieren eine Verbindung zu den jeweiligen CSV-Dateien und die Möglichkeit Daten aus diesen Dateien auszulesen. Dafür wird die Datei geladen und jede Zeile repräsentiert die Informationen eines individuellen Objekts. Die jeweiligen Spalten stellen die zugehörigen Attribute dar. Um ein verwertbares Objekt zu erstellen, werden die entsprechenden Informationen einer Zelle auf die Attribute eines C#-Objekts gemappt. Entsprechend werden Änderungen gespeichert, indem Attributwerte eines C#-Objekts in entsprechende Zellen der CSV-Datei geschrieben werden. Welche Objekte genau

aus der Datenbasis geladen werden, wird über LINQ-Ausdrücke definiert. LINQ ermöglicht es im Programmcode, SQL-ähnliche Abfragen an eine Datenbasis zu senden. So ist es beispielsweise möglich, alle Objekte einer Tabelle zu laden, oder nur solche, die eine bestimmte Bedingung erfüllen. Mit dem Ausdruck „FirstOrDefault“ wird beispielsweise das erste Tier aus der Datenbank gelesen, welches eine übereinstimmende ID mit der Abfrage hat. Da jede ID einzigartig ist, entspricht das erste Tier mit einer bestimmten ID gleichzeitig dem einzigen Tier mit einer bestimmten ID.

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

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

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

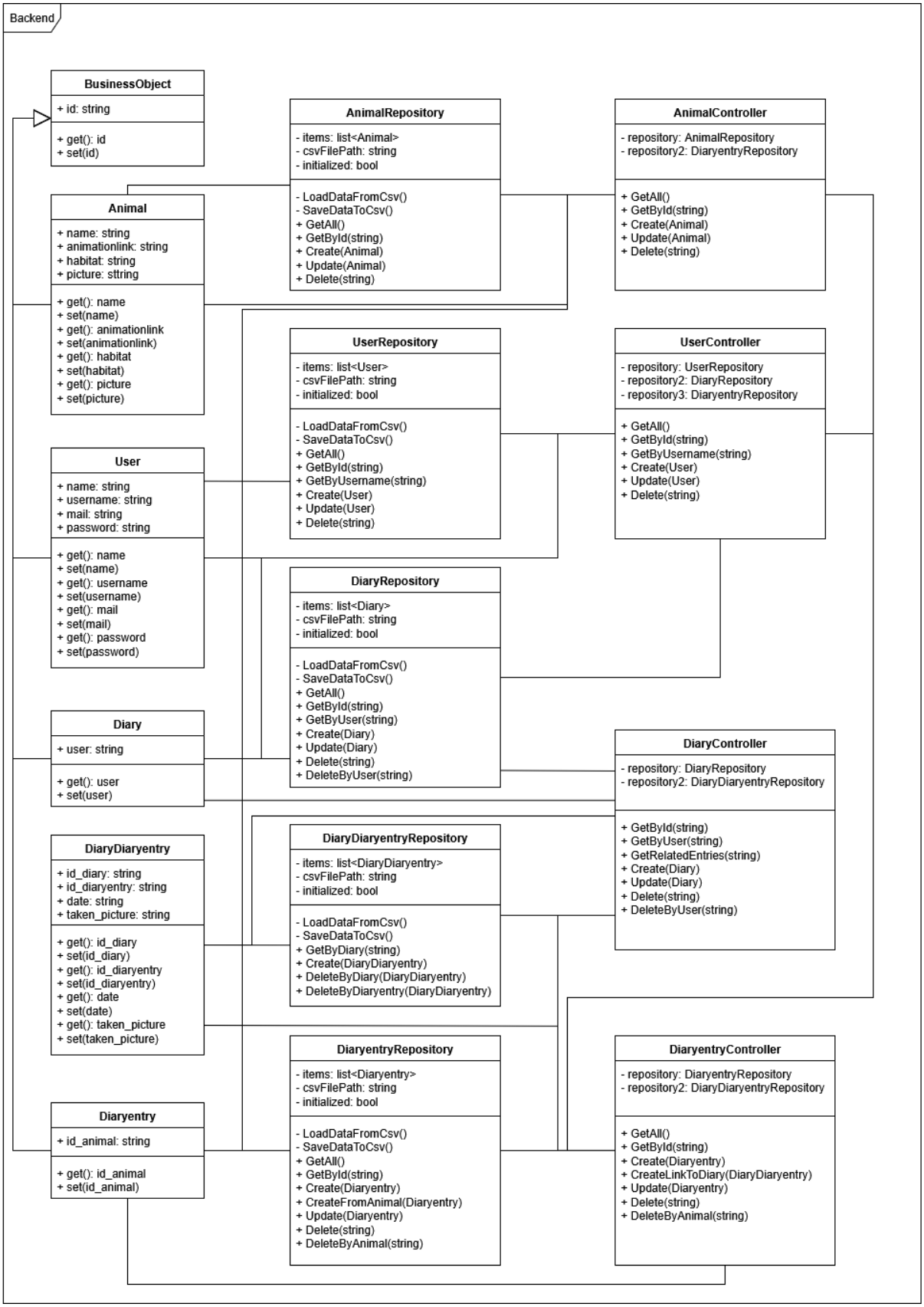

Abbildung 8 - UML-Diagramm Backend (eigene Darstellung)

6.3 Frontend

6.3.1 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 (Unity Technologies, o. J.-b). Dazu zählen das Android Build Support Modul, das Android Software Development Kit (SDK), das Native Development Kit (NDK) sowie das Java Development Kit (OpenJDK), das standardmäßig von Unity verwendet wird. Diese Komponenten sind für die Ausführung der Anwendung auf der Meta Quest 3 notwendig (Unity Technologies, o. J.-a).

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 Übertragung der Anwendung auf die Meta Quest 3 setzt die Aktivierung des Entwicklermodus voraus. Zuerst wird das Headset mit einem Meta-Konto verknüpft. Anschließend wird der Entwicklermodus über die Meta Horizon App auf einem verbundenen Smartphone aktiviert (Meta, 2025c).

Die Übertragung der Anwendung auf die Meta Quest 3 erfolgt über die „Build and Run“-Funktion erfolgen. 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 schnell umgesetzt werden (Meta, 2025c).

6.3.2 Implementierung

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

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

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

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

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

Die View-Schicht ist für die Verknüpfung von UI-Elementen und den dahinterliegenden Funktionen im ViewModel zuständig. Damit ist diese Schicht die Einzige, die vollständig unityspezifisch ist. Wie bereits erläutert, kann über den Inspector beispielsweise ein Button mit

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

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

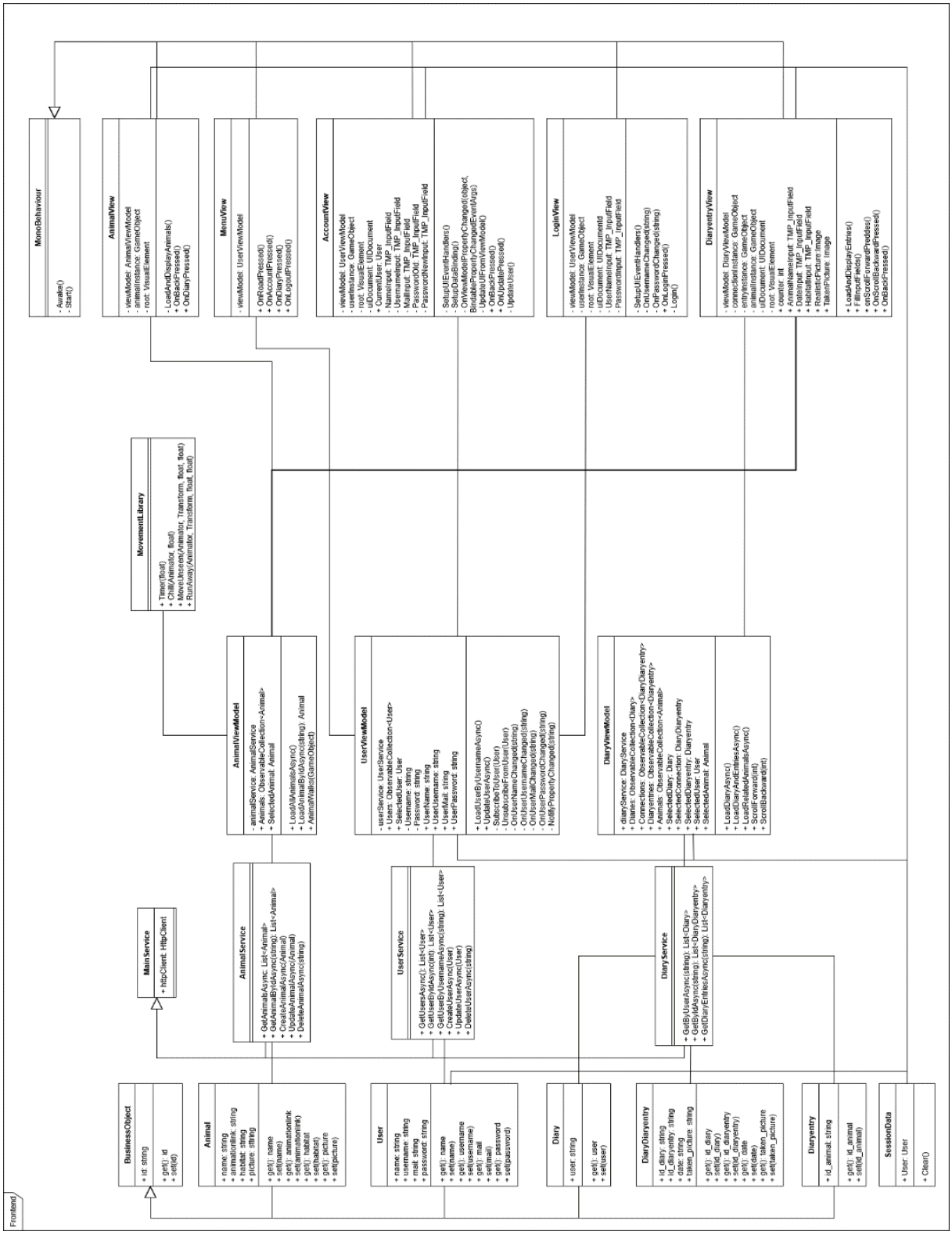

Abbildung 9 - UML-Diagramm Frontend (eigene Darstellung)

Mit diesem Aufbau sind Verantwortlichkeiten nach dem MVVM- beziehungsweise MVVM-3D-Muster getrennt. Zentral für das Zusammenspiel zwischen View und ViewModel, und damit der zentrale Vorteil dieser Architektur, ist das Data Binding. Wie im Kapitel 4.4 Architekturmodell erläutert, ermöglicht es automatische Änderungen an der UI bei Änderungen an Properties und umgekehrt die direkte Übernahme von Benutzereingaben in das ViewModel. Nachfolgend wird die Umsetzung von Data Binding in Unity am Beispiel „Passwort ändern“ erläutert, da jenes elementar für die Funktion der weiterentwickelten Architektur ist.

Für den Prototypen wurde das Data Binding mittels UI Toolkit und einer Implementierung mit Properties und Events realisiert. Die relevanten Felder im User-Model sind mit „SerializeField“ und „Serializable“ versehen, damit Unity sie serialisieren und in der graphischen Oberfläche anzeigen, weiter mit Funktionen oder Variablen verknüpfen kann. Für die Synchronisierung zwischen View und ViewModel verwenden die ViewModels ObservableCollections und implementieren das Interface INotifyBindablePropertyChanged. Über dieses Interface können Änderungen an Properties sofort als Event gemeldet und daraufhin die UI aktualisiert werden. Das Speichern eines Passworts verdeutlicht diesen Ablauf. Die UserView enthält zwei Eingabefelder – eines für das aktuelle Passwort und eines für die Eingabe eines neuen Passworts. Beim Start der Szene werden diese Felder über die Start()-Methode mit entsprechenden Properties im ViewModel verknüpft. Diese Verknüpfung erfolgt über onValueChanged-Listen.

Damit wird bei einer Eingabe diese direkt an die passende Property im ViewModel übergeben. Dort löst die Property ein Änderungs-Event aus, wodurch wiederum alle gebundenen UI-Elemente automatisch aktualisiert werden. So erscheint eine Passwortänderung sofort auch im Feld für das aktuelle Passwort. Damit die Änderung am Account nicht nur in der aktuellen Szene, sondern im gesamten Spiel wirksam wird, muss sie explizit gespeichert werden. Dies geschieht über einen Button, der eine Übertragung der Daten an das Backend auslöst.

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

6.3.3 Weiteres

Einen Überblick über weitere relevante Implementierungen bieten die nachfolgenden Unterkapitel:

Animationen Da für die virtuelle Tierbeobachtung ein realistisches Verhalten von virtuellen Tieren vorausgesetzt wird, werden 3D-Modelle mit animierten Komponenten verwendet. Alle verwendeten 3D-Tiermodelle stammen von Sketchfab (Sketchfab, 2025). Für das Projekt wurden ausschließlich kostenlose Modelle genutzt, da viele hochwertige, realistisch animierte Tiere kostenpflichtig sind und die Preise je nach Modell von wenigen Euro bis zu mehreren hundert Euro reichen können. Die Auswahl geeigneter Modelle gestaltete sich insofern anspruchsvoll, als viele verfügbare Modelle entweder statisch oder auf eine einzelne Bewegung

beschränkt waren. Für den Prototyp konnten jedoch Modelle identifiziert werden, die den definierten Anforderungen entsprechen und eine realistische Abbildung von Tierbewegungen in der virtuellen Umgebung ermöglichen. Die zugehörigen Animationsclips umfassen Aktionen wie beispielsweise Laufen, Ablegen oder Rennen. Allerdings sind die einzelnen Animationen nur wenige Sekunden lang und längere Bewegungsabläufe können nur dann erstellt werden, wenn sie mit mehreren (unterschiedlichen) Animationen zusammengesetzt werden.

Als erste Umgebung wird ein Wald gewählt, da dieser Lebensraum weit verbreitet ist, eine hohe Artenvielfalt aufweist und sich gut für die erste Testumgebung eignet. Als erstes Testtier wird ein Wapiti ausgewählt, da dieses Tier sehr schön animiert ist, gut ins Waldszenario passt und viele verschiedene Bewegungen bietet. Darüber hinaus wurden bereits weitere Tiere mit guten Animationen, wie Bär, Luchs, verschiedene Vögel und Schmetterlinge, herausgesucht. Diese können später in den Prototypen integriert werden. Für erste Tests soll der Prototyp jedoch zunächst mit einer Umgebung und einem Tier erprobt werden. Für zukünftige Erweiterungen der Strecken wurden zudem bereits Modelle anderer Lebensräume und Tiere heruntergeladen, z. B. Savanne mit Elefant, Zebra oder Krokodil sowie ein Nachtmodus mit Eule und Fledermaus, falls diese später noch integriert werden sollten.

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

[12] Ein Controller in diesem Kontext bezieht sich auf die Steuerung von Animationen und nicht auf die Steuerung zwischen Model und View, wie er im Kontext der MVC-Architektur verstanden wird. Der Controller hat an dieser Stelle keine Verbindung zum Architekturmodell.

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

Interaktion

Die Interaktion im Prototypen wurde mit Hilfe des Meta All-in-One SDK umgesetzt. Dieses stellt mehrere verschiedene sogenannte Building Blocks zur Verfügung. Die Module bieten vorgefertigte, leicht integrierbare Funktionen wie Controllersteuerung, Handtracking, Passthrough oder kollisionsfähige Meshes und ermöglichen dadurch eine deutliche Zeitersparnis bei der Entwicklung. Anstatt grundlegende Interaktionsmechanismen selbst implementieren zu müssen, können etablierte, bereits optimierte Bausteine genutzt werden, die speziell für die Meta Quest 3 konzipiert sind (Dev, 2024). Dadurch entsteht eine stabile Basis, auf der spezifische Logiken und Features des Projekts aufbauen können.

Für die Darstellung der Umgebung kommt das Mixed Reality Utility Kit (MRUK) zum Einsatz, das in Kombination mit dem Effect Mesh ein statisches, kollisionsfähiges Mesh erzeugt. Dieses Mesh bildet Wände, Boden und Hindernisse eines einzelnen Raumes ab und stellt sicher, dass die virtuellen Tiere auf physikalisch sinnvolle Weise mit der Umgebung interagieren können. Ein kontinuierliches Scannen der realen Umgebung wurde bewusst vermieden, da dies sowohl hohe Rechenressourcen als auch Speicherplatz beanspruchen würde. Stattdessen erlaubt das statische Mesh eine stabile Performance und eine verlässliche Kollisionslogik für den Prototyp. Ergänzend wurde auch der Passthrough-Building Block eingebunden, wodurch reale Elemente sichtbar bleiben und eine bessere Verankerung der virtuellen Inhalte in der Realität entsteht.

Für die Steuerung werden sowohl Controller-Interaktionen als auch Handtracking eingesetzt. Beide Varianten sind über Building Blocks integriert und bieten den Nutzenden Flexibilität. Während Controller präzise Eingaben ermöglichen, schafft Handtracking ein intuitives, natürliches Bediengefühl über Gesten wie Zeigen, Greifen oder Wischen. Ein zentrales technisches Problem besteht derzeit darin, dass auf der MR-Brille Buttons und Eingabefelder nicht anwählbar sind. Der sogenannte Ray, der üblicherweise als visuelle Markierung für Auswahl und Interaktion dient, wird nicht angezeigt. Trotz mehrfacher Anpassungen der Konfigurationen konnte dieses Problem bislang nicht behoben werden. Um dennoch erste Interaktionen überprüfen zu können, wurde in der Login-Szene ein Würfel als Hilfsobjekt integriert, der sich sowohl über die Controller als auch mittels Handtracking greifen lässt. Auf diese Weise konnte die grundlegende Funktionsfähigkeit von Interaktionen mit beiden Eingabemethoden bestätigt werden. Eine stabile und konsistente UI-Steuerung ist damit jedoch noch nicht gewährleistet, sodass ein Wechsel zwischen Szenen derzeit nicht möglich ist.

Dadurch konnten weiterführende Mechanismen wie die geplante Interaktion mit den virtuellen Tieren oder deren Fluchtverhalten bislang noch nicht praktisch umgesetzt werden.

Ein nächster Schritt wäre die Implementierung einer virtuellen Tastatur, die automatisch erscheint, sobald ein Eingabefeld angewählt wird. Dadurch könnten Nutzende auch direkt auf der Brille Texteingaben vornehmen, beispielsweise für die Anmeldung oder das digitale Entdeckertagebuch. Im Unity-Editor funktioniert die Texteingabe bereits mit Maus und normaler Tastatur, doch die Übertragung auf die Quest-Hardware stellt noch eine Hürde dar.

Für die Zukunft eröffnet die derzeitige Architektur weitere Möglichkeiten. So könnten Interaktionen wie Zeigen, Greifen oder das Auslösen von UI-Elementen auch in einem Multiplayer-Kontext synchronisiert werden, sodass mehrere Nutzende gleichzeitig dieselbe Szene erleben und gemeinsam mit der virtuellen Umgebung interagieren. Aktuell liegt der Fokus jedoch auf der stabilen Realisierung grundlegender Interaktionen in einem vordefinierten Raum.

Zusammenfassend bildet die Nutzung der Meta-Building-Blocks eine solide Grundlage für die Interaktion im Prototypen. Während grundlegende Mechanismen wie Hand- und Controllersteuerung sowie die Mesh-Umgebung bereits funktionsfähig implementiert sind, bestehen noch Herausforderungen bei der direkten UI-Nutzung auf der Brille. Mit der geplanten Integration einer virtuellen Tastatur und der Behebung der Ray-Problematik sollen diese Hürden in den nächsten Iterationen adressiert werden, um ein vollständig immersives und benutzerfreundliches Interaktionserlebnis zu ermöglichen.

7 Zusammenfassung und Ausblick

Die vorliegende Arbeit behandelt die Entwicklung eines Mixed-Reality-Prototyps für die virtuelle Tierbeobachtung auf der Meta Quest 3. Ziel war die Umsetzung einer immersiven, interaktiven Anwendung, die sowohl das Beobachten und Fotografieren als auch die Interaktion mit virtuellen Tieren möglich macht.

Die Basisarchitektur der Anwendung folgt dem MVVM-3D-Muster, das die klassischen Schichten Model, View und ViewModel um eine Library-Schicht erweitert. Models enthalten die Datenstruktur, während Services die Schnittstelle zum Backend übernehmen. ViewModels steuern die Logik, verarbeiten Eingaben und synchronisieren automatisch die Anzeige über Data Binding. Dadurch werden Änderungen direkt in der View sichtbar. Dies erhöht die Reaktionsfähigkeit des Systems, was besonders für Fotografie und Echtzeitinteraktionen relevant ist. Zudem enthält die View Komponenten, die multimodale Eingaben möglich machen und das System immersiver gestalten. Die Library kapselt komplexe Berechnungen, Bewegungsabläufe der Tiere und Unity-spezifische Funktionen. Frontend und Backend wurden in Unity/C# bzw. ASP.NET Core umgesetzt, die Datenbasis zunächst über lokal gespeicherte CSV-Dateien realisiert.

Die Controller ermöglichen präzise Eingaben und Navigation, das Handtracking erlaubt natürliche Gesten, und Texteingaben dienen Login und Personalisierung. Die Umgebung wird durch ein vordefiniertes, kollisionsfähiges Mesh abgebildet, das realistische Tierreaktionen wie Flucht oder Wegfindung unterstützt. Fotografie kann über Controller ausgelöst werden und wird automatisch im digitalen Entdeckertagebuch gespeichert. Building Blocks des Meta SDK, darunter MRUK, Effect Mesh und Passthrough, erleichtern die Integration von Eingabemethoden, Kollisionslogik und Mixed-Reality-Funktionen.

Einschränkungen bestehen derzeit bei der Bedienung von UI-Elementen auf der Brille, da der Ray für Eingabefelder und Buttons nicht sichtbar ist. Die Implementierung einer virtuellen Tastatur ist geplant, sobald die Auswahl der Felder zuverlässig funktioniert.

Für zukünftige Erweiterungen bietet das Konzept zahlreiche Möglichkeiten. Tiere könnten durch qualitativ bessere Animationen, zum Beispiel selbst in Blender erstellt oder in Zusammenarbeit mit weiteren Studierenden anderer Fachrichtungen entwickelt, realistischer dargestellt werden. Eine kontinuierliche Anpassung an die reale Umgebung durch dynamisches Raumscanning könnte die Immersion erhöhen. Multiplayer-Interaktionen, in denen mehrere Nutzende gleichzeitig die gleiche Szene erleben und Tiere gemeinsam beobachten oder Aktionen synchron ausführen, bieten ebenfalls großes Potential. Die modulare MVVM-3D-Struktur ermöglicht zudem die einfache Integration neuer Tierarten, Interaktionsmöglichkeiten oder Erweiterungen. Eine schnellere Reaktionsfähigkeit durch Data Binding eröffnet ebenfalls neue Möglichkeiten bezüglich einer besseren Immersion dank einer unmittelbaren Rückkopplung zwischen Nutzereingaben und Systemreaktionen.

Insgesamt zeigt der Prototyp, wie Mixed-Reality-Anwendungen sowohl immersiv als auch interaktiv gestaltet werden können, und legt eine flexible, erweiterbare Grundlage für zukünftige Entwicklungen in den Bereichen Bildung, Forschung, Unterhaltung und Erlebnisse mit virtuellen Tieren.

Anhang A **Zukünftige Forschungsfragen zu XR**

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

- Inhalte
  - Nützlich (Utilitarian)
    - Welchen tatsächlichen Effekt hat nützlicher AR-Content auf die kognitive Verarbeitung von Konsument:innen im Rahmen des Zwei-Prozess-Modells der Informationsverarbeitung?
    - Welche Gestaltungsprinzipien können die kognitive Verarbeitung in AR optimieren, um die kognitive Belastung zu reduzieren?
    - Wie beeinflussen Unterschiede in visuellen Details, Farbe und Kontrast von AR-Inhalten kognitive Prozesse wie die Aufmerksamkeit während Entscheidungsfindungen?
  - Hedonistisch (Hedonic)
    - Wie beeinflussen unterhaltsame Elemente in AR-Inhalten die Emotionen von Konsument:innen? Wie entwickeln sich diese Emotionen über die Dauer einer längeren AR-Erfahrung?
    - Welche Arten von hedonistischem Content in AR (z. B. Gamification, interaktive Erzählformen) sind am effektivsten, um positive emotionale Reaktionen hervorzurufen?
    - Wie wirkt sich unterhaltsamer AR-Content auf das Wohlbefinden der Konsument:innen aus (z. B. Stressniveau, Stimmung, Glück)?
  - Sozial (Social)
    - Wie beeinflusst die Integration sozialer Elemente (z. B. Sharing-Funktionen) die wahrgenommene Glaubwürdigkeit und Vertrauenswürdigkeit der präsentierten Informationen?
    - Wie wirken sich unterschiedliche soziale Elemente (z. B. virtuelle Avatare, Echtzeit-Interaktionen) auf das Konsumentenverhalten aus?
    - Welche Rolle spielen Empfehlungen von Gleichgesinnten und sozialer Beweis (social proof) bei Konsumentscheidungen?
  - Generative KI (GenAI)
    - Welchen Einfluss hat individuell anpassbarer GenAI-Content auf das Engagement der Konsument:innen?
    - Wie unterscheiden sich die Reaktionen der Konsument:innen auf GenAI-Content im Vergleich zu manuell erstelltem Content in AR-Umgebungen?
    - Wie kann KI-gesteuerte, echtzeitbasierte Anpassung an die Umgebung in AR die Nutzererfahrung verbessern?
- Kontext
  - Produkttyp
    - Wie beeinflusst AR das Vertrauen und die Entscheidungsfindung von Konsument:innen bei risikoreichen Produkten im Vergleich zu mittel- und niedrig-risikohaften Produkten?
    - Wie helfen AR-Funktionen dabei, wahrgenommenes Risiko zu verringern und Vertrauen bei risikoreichen Produkten zu stärken?
    - Wie beeinflusst die Vertrautheit der Konsument:innen mit einem Produkt deren Reaktion auf AR-unterstützte Produktpräsentationen?
  - Marken

        - Wie wirkt sich AR auf die Markenwahrnehmung und -bindung bei lokalen im Vergleich zu globalen Marken aus?
        - Wie beeinflusst der Einsatz von AR die Wahrnehmung und Einstellung der Konsument:innen gegenüber bekannten versus unbekannten Marken?
        - Welche Unterschiede zeigen sich in den emotionalen Reaktionen auf AR-Erlebnisse bei der Interaktion mit vertrauten vs. unbekannten Marken?
    - Umgebung
        - Wie beeinflussen multisensorische AR-Erfahrungen (z. B. Kombination aus visuellen, auditiven und olfaktorischen Reizen) die Bewertung und das Verhalten der Konsument:innen?
        - Wie wirkt sich die wahrgenommene Privatsphäre der Umgebung auf die Bereitschaft der Konsument:innen aus, persönliche Informationen zu teilen und mit AR-Features zu interagieren?
        - Wie beeinflusst die Kongruenz des AR-Raumes (z. B. virtuelle Produkte in passenden Umgebungen) die Wahrnehmung von Produktqualität und Relevanz?
    - Business-to-Business (B2B)
        - Wie beeinflusst der Einsatz von AR in Mitarbeiterschulungen Lernresultate, Wissenserhalt und Arbeitsleistung im Vergleich zu traditionellen Schulungsmethoden?
        - Wie wirkt sich AR-gestützte Remote-Zusammenarbeit auf Entscheidungsfindung, Projekteffizienz und Markteinführungszeiten in Produktentwicklungsteams aus?
        - Welche Auswirkungen haben virtuelle AR-Ausstellungen auf Kund:innenbindung und Conversion-Rates im Vergleich zu traditionellen Marketingansätzen?
- Gerät (Device)
    - Wie unterscheiden sich Bedienbarkeit und Benutzerfreundlichkeit von AR-Apps auf mobilen Geräten, Computern und AR-Brillen, und wie wirkt sich das auf die Akzeptanz bei Konsument:innen aus?
    - Wie beeinflussen virtuelle Spiegel die Wahrnehmung von Passform, Qualität und Kaufabsicht im Vergleich zu traditionellen Einkaufsmethoden?
    - Welche kognitiven und emotionalen Unterschiede ergeben sich bei der Nutzung von AR über mobile Geräte versus AR-Brillen?
- Konsument:innen
    - Soziodemografische Aspekte
        - Welche spezifischen AR-Funktionen sprechen jüngere im Vergleich zu älteren Konsument:innen besonders an?
        - Wie kann AR-Content kulturell vielfältig angepasst werden?
        - Welche Strategien können genutzt werden, um AR-Apps benutzerfreundlicher für Konsument:innen mit unterschiedlichen technischen Fähigkeiten zu gestalten?
    - Ethische Bedenken
        - Wie nehmen Konsument:innen die ethische Vertretbarkeit emotionaler Beeinflussung durch AR im Marketing und Einzelhandel wahr?
        - Wie wird die ethische Frage bewertet, dass AR zu Impulskäufen verleiten kann?
        - Welche Strategien können helfen, Datenschutzbedenken zu verringern und Vertrauen in AR-Technologien aufzubauen?
- Methodik
    - Messmethoden

        - Wie können physiologische Messmethoden wie Eye-Tracking und Herzfrequenzmessung zur Bewertung von Aufmerksamkeit und emotionalen Reaktionen in AR-Erfahrungen eingesetzt werden?
        - Wie beeinflussen AR-Erfahrungen Konsumentenverhalten, z. B. Kaufentscheidungen oder Rückgabequoten?
        - Wie entwickeln sich Konsumentenwahrnehmung und -nutzung von AR-Technologie bei wiederholtem Kontakt?
    - Forschungsmethoden
        - Wie lassen sich Langzeitstudien gestalten, um Veränderungen im Konsumentenverhalten und in der Einstellung zu AR nachzuvollziehen?
        - Welche spezifischen Muster lassen sich durch Mixed-Methods-Forschung (Kombination von qualitativen und quantitativen Daten) identifizieren?

Wie unterscheiden sich AR-Erlebnisse in natürlichen Umgebungen von solchen in kontrollierten Laborsituationen hinsichtlich der Konsumentenreaktionen?

Abbildungsverzeichnis

Tabellenverzeichnis

Software Tools

Git	Git – https://git-scm.com/
GitHub	GitHub – https://github.com/
Swagger UI	Swagger – https://swagger.io/tools/swagger-ui/
Unity 6.0	Unity – https://unity.com
Unity Hub	Unity – https://unity.com/unity-hub
Visual Studio 2022	Microsoft – https://visualstudio.microsoft.com/

Literaturverzeichnis

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/

Awati, R. (2024, Januar 22). *What is 3D mesh? | Definition from TechTarget*. WhatIs. https://www.techtarget.com/whatis/definition/3D-mesh

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.

Berman, M. G., Kross, E., Krpan, K. M., Askren, M. K., Burson, A., Deldin, P. J., Kaplan, S., Sherdell, L., Gotlib, I. H., & Jonides, J. (2012). Interacting with nature improves

cognition and affect for individuals with depression. *Journal of Affective Disorders, 140*(3), 300–305. https://doi.org/10.1016/j.jad.2012.03.012

Bezmalinovic, T. (2022, Februar 26). National Geographic VR im Test: VR-Reisen für Quest (2). *MIXED*. https://mixed.de/national-geographic-vr-test/

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

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

BRINK XR. (o. J.). Abgerufen 26. August 2025, von https://www.BrinkXR.com/

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

Carter, R. (2024, November 29). What is VR Passthrough and How is it Shaping the Future of XR? *XR Today*. https://www.xrtoday.com/virtual-reality/what-is-vr-passthrough-and-how-is-it-shaping-the-future-of-xr/

Circuit Stream. (2022, Mai 22). *Circuit Stream · C# vs C++: Complete Comparison Between Unity and Unreal Programming Language*. https://www.circuitstream.com/en/blog/c-vs-c-complete-comparison-between-unity-and-unreal-programming-language

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/

Dev, S. (2024, Februar 15). Exploring Meta's Building Blocks. *Antaeus AR*. https://medium.com/antaeus-ar/exploring-metas-building-blocks-d99fdd1b3ec3

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

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

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

Erl, J., & Danneberg, B. (2025, April 1). *VR-Brillen 2025: Vergleich & Kaufberatung – Das müsst ihr wissen*. https://mixed.de/vr-brillen-vergleich/

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

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

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

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

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

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

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

IMARC Group. (o. J.). *Head-Mounted Displays (HMDs): A Look into Immersive Technologies*. Abgerufen 29. August 2025, von https://www.imarcgroup.com/insight/head-mounted-displays-hmds-a-look-into-immersive-technologies

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

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

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

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

Lorenz-Spreen, P., Mønsted, B. M., Hövel, P., & Lehmann, S. (2019). Accelerating dynamics of collective attention. *Nature Communications*, *10*(1), 1759. https://doi.org/10.1038/s41467-019-09311-w

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/

Max-Planck-Institut für Bildungsforschung. (2025, April 15). *Mit der Informationsflut sinkt die Aufmerksamkeitsspanne der Gesellschaft*. https://www.mpib-berlin.mpg.de/pressemeldungen/informationsflut-senkt-aufmerksamkeitsspanne

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. (o. J.). *Gesundheits- und Sicherheitshinweise | Meta Quest 3 | Meta Store*. Abgerufen 30. August 2025, von https://www.meta.com/de/legal/quest/health-and-safety-warnings/quest-3/?srsltid=AfmBOooPETPtFKDh3HdagR_8ydlVfx8j-LseEdJudS5afWLpXxkgjN4U.com

Meta. (2025a). *Animalz Mixed Reality auf Meta Quest*. Oculus. https://www.meta.com/de-de/experiences/animalz-mixed-reality/5583019775107608/

Meta. (2025b). *BRINK Traveler auf Meta Quest*. Oculus. https://www.meta.com/de-de/experiences/brink-traveler/3635172946605196/

Meta. (2025c). *Dein Gerät einrichten*. https://developers.meta.com/horizon/documentation/unity/unity-env-device-setup

Meta. (2025d). *Living Room auf Meta Quest*. Oculus. https://www.meta.com/de-de/experiences/living-room/7778145568911617/

Meta. (2025e). *National Geographic Explore VR für Meta Quest | Quest VR-Games | Meta Store*. https://www.meta.com/de-de/experiences/national-geographic-explore-vr/2046607608728563/?srsltid=AfmBOorM2xef3Rcf918a1jHmO7uqH9ncTdd5dBgzzV7yaCc8soghS5ln

Meta. (2025f). *Nature Treks VR für Meta Quest | Quest VR-Games | Meta Store*. https://www.meta.com/de-de/experiences/nature-treks-vr/2616537008386430/

Meta. (2025g). *Ocean Rift auf Meta Quest*. Oculus. https://www.meta.com/de-de/experiences/ocean-rift/2134272053250863/

Meta. (2025h). *WildXR für Meta Quest | Quest VR-Games | Meta Store*. https://www.meta.com/de-de/experiences/wildxr/3634799699973926/

Meta. (2025i). *ZOSU Zoo auf Meta Quest*. Oculus. https://www.meta.com/de-de/experiences/zosu-zoo/3623396967784471/

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

Picselica. (o. J.). *Ocean Rift*. Picselica. Abgerufen 26. August 2025, von https://www.picselicavr.com/oceanrift

Rambach, J., Lilligreen, G., Schäfer, A., Bankanal, R., Wiebel, A., & Stricker, D. (2020). *A survey on applications of augmented, mixed and virtual reality for nature and environment* (Version 2). arXiv. https://doi.org/10.48550/ARXIV.2008.12024

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/

Ratinho, E., & Martins, C. (2023). The role of gamified learning strategies in student's motivation in high school and higher education: A systematic review. *Heliyon*, *9*(8), e19033. https://doi.org/10.1016/j.heliyon.2023.e19033

Rees, C. (2025, März 13). *VR Software wiki—Meta Building Blocks in Unity*. https://www.vrwiki.cs.brown.edu/vr-development-software/unity/meta-building-blocks-in-unity

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

Seydel. (o. J.). *Head Mounted Display—5 Einsatzgebiete garantiert*. Abgerufen 28. August 2025, von https://triboot.de/head-mounted-displays/

Sketchfab. (2025). *Animals & Pets 3D models | Categories*. Sketchfab. https://sketchfab.com/3d-models/categories/animals-pets?features=downloadable+animated&sort_by=-likeCount&cursor=cD03MzE%3D

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

Streule, N. (o. J.). Living Room. *Thoughtfish*. Abgerufen 26. August 2025, von https://www.thoughtfish.de/projects/living-room/

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. (o. J.-a). *Unity - Manual: Android environment setup*. Abgerufen 20. August 2025, von https://docs.unity3d.com/6000.0/Documentation/Manual/android-sdksetup.html

Unity Technologies. (o. J.-b). *Unity - Manual: Install Unity*. Abgerufen 20. August 2025, von https://docs.unity3d.com/6000.0/Documentation/Manual/GettingStartedInstallingUnity.html

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

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

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

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

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

Wilczyńska, D., Walczak-Kozłowska, T., Alarcón, D., Arenilla, M. J., Jaenes, J. C., Hejła, M., Lipowski, M., Nestorowicz, J., & Olszewski, H. (2024). The Role of Immersive Experience in Anxiety Reduction: Evidence from Virtual Reality Sessions. *Brain Sciences*, *15*(1), 14. https://doi.org/10.3390/brainsci15010014

William, R. (2023, Juni 11). *Inside-Out vs Outside-In VR Tracking: The Ultimate Guide*. https://arvrtips.com/inside-out-outside-in-vr-tracking/

Wölfel, M. (2023). *Immersive Virtuelle Realität: Grundlagen, Technologien, Anwendungen*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-66908-2

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

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

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

Zobel, B., Werning, S., Metzger, D., & Thomas, O. (2018). Augmented und Virtual Reality: Stand der Technik, Nutzenpotenziale und Einsatzgebiete. In C. de Witt & C. Gloerfeld

(Hrsg.), *Handbuch Mobile Learning* (S. 123–140). Springer Fachmedien. https://doi.org/10.1007/978-3-658-19123-8_7

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