HTBL Krems

Höhere Lehranstalt für Informationstechnologie
Ausbildungsschwerpunkt: Medientechnik

DIPLOMARBEIT

WoSamma

Ausgeführt im Schuljahr 2025/26 von:

Jan Tiefenbacher	5BHITM-01
Laurenz Pichler	5BHITM-02

Betreuer:

Dipl.-Ing. (FH) Brandstetter Gerald
Dipl.-Ing. (FH) Brandstetter Gerald

Krems, am 03.04.2026

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Krems, am 03. April 2026

Verfasser/Verfasserinnen:

Jan Tiefenbacher

Laurenz Pichler

DIPLOMARBEIT

Bestätigung der Abgabe

Abgabebestätigung

Datum	Name	Unterschrift

Genehmigung der Diplomarbeit

Approbation

Datum	Prüfer*in	Abteilungsleiter*in Direktor*in

DIPLOMARBEIT

Dokumentation

Verfasser*innen
Jan Tiefenbacher, 5BHITM
Laurenz Pichler, 5BHITM
Abteilung
Informationstechnologie
Ausbildungsschwerpunkt: Medientechnik
Schuljahr
2025/26
Thema der Diplomarbeit
Wosamma geoinformationsbasiertes Quizzspiel
Kooperationspartner
Xinger Solutions GmbH

Aufgabenstellung
Ziel des Projekts WoSamma war die Entwicklung einer mobilen, geoinformationsbasierten Quiz-App, die sich auf Österreich fokussiert. Spieler sollen anhand von Street-View-Bildern erraten, an welchem Ort sie sich befinden, und dafür Punkte erhalten. Die App soll sowohl Lern- als auch Unterhaltungszwecke erfüllen und österreichisches Geografiewissen spielerisch vermitteln. Neben der Einzelspieler-Funktion war auch die Umsetzung von Mehrspielermodi, Ranglisten, Freundesystemen und einem Administrationsbereich geplant. Die Anwendung sollte plattformunabhängig lauffähig sein und insbesondere auf mobilen Endgeräten eine flüssige Nutzererfahrung bieten.

Realisierung

- **Programmiersprache / Framework:** React Native mit TypeScript
- **Backend & Datenbank:** Supabase (PostgreSQL, Auth, Storage)
- **Deployment:** Mit Apple Developer Konto auf dem Handy nutzbar
- **UI/UX-Design:** Figma (Mockups & Komponenten)
- **Versionsverwaltung:** Git & GitHub
- **Projektmanagement:** Jira (agiles Vorgehen, Sprints)
- **Besonderheiten bei der Entwicklung:**
  - Performance-Optimierung für mobile Geräte
  - API-Sicherheit & Authentifizierung
  - Plattformübergreifende Kompatibilität (iOS & Android)
  - Echtzeitfunktionen (Chat, Multiplayer)

Ergebnisse

Eine voll funktionsfähige Mobile-App mit vielfältigen Spielmodi, integriertem Freundesystem und Echtzeit-Chat für ein dynamisches und interaktives Spielerlebnis.

DIPLOMA THESIS

Documentation

Authors
Jan Tiefenbacher, 5BHITM
Laurenz Pichler, 5BHITM
Department
Informationstechnologie
Specialization: Medientechnik
Academic year
2025/26
Thesis Topic
WoSamma geo-information-based quiz game
Co-operation partners
Xinger Solutions GmbH

Task Description
The goal of the WoSamma project was to develop a mobile, geo-information-based quiz app focused on Austria. Players should guess their location based on Street View images and earn points for correct answers. The app is designed to be both educational and entertaining, helping users learn about Austrian geography in a playful way.
In addition to a single-player mode, the project also included the implementation of multiplayer modes, leaderboards, friend systems, and an administration area. The application was intended to run on multiple platforms and provide a smooth user experience, especially on mobile devices.

Implementation

- **Programming Language / Framework:** React Native with TypeScript
- **Backend & Database:** Supabase (PostgreSQL, Auth, Storage)
- **Deployment:** Usable on mobile devices via an Apple Developer account
- **UI/UX Design:** Figma (mockups & components)
- **Version Control:** Git & GitHub
- **Project Management:** Jira (agile workflow, sprints)
- **Special Aspects of Development:**
  - Performance optimization for mobile devices
  - API security & authentication
  - Cross-platform compatibility (iOS & Android)
  - Real-time features (chat, multiplayer)

Results

A fully functional mobile app featuring multiple game modes, an integrated friends system, and real-time chat for a dynamic and interactive player experience.

Inhaltsverzeichnis

1. Präambel

1.1. Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde die mobile Anwendung WoSamma entwickelt – ein geoinformationsbasiertes Quizspiel, das es den Nutzern ermöglicht, Orte innerhalb Österreichs zu erkennen. Die App bietet verschiedene Spielmodi, darunter Einzelspieler-, Freundes- und Mehrspielervarianten. Zusätzlich wurden ein Freundes- und Chatsystem integriert, um den sozialen Aspekt des Spiels zu fördern.

- React Native App (mit TypeScript & Expo)
- Supabase (PostgreSQL, Authentifizierung, Storage)
- Eigene REST-API für Datenkommunikation

1.2. Abstract

As part of this diploma thesis, the mobile application WoSamma was developed – a geo-information-based quiz game that allows users to recognize and locate virtual places within Austria. The app combines playful learning with modern mapping technology and offers various game modes, including single-player, friends, and multiplayer modes. In addition, a friends and chat system was integrated to enhance the social aspect of the game.

- React Native App (with TypeScript & Expo)
- Supabase (PostgreSQL, Authentication, Storage)
- Custom REST API for data communication

1.3. Team

Das Projektteam besteht aus dem Projektleiter Jan Tiefenbacher und Kollegen Laurenz Pichler. Betreuer ist DI(FH) Brandstetter Gerald, der Auftraggeber ist Xinger Solutions GmbH.

1.4. Danksagung

Unser besonderer Dank gilt Dipl.-Ing. (FH) Gerald Brandstetter für seine engagierte und kompetente Betreuung im Rahmen dieser Diplomarbeit. Er unterstützte uns insbesondere im Bereich des UI-Designs mit wertvollen Anregungen und fachlicher Expertise. Auch im Backend-Bereich trug sein umfangreiches Know-how wesentlich zur erfolgreichen Umsetzung bei. Seine kontinuierliche Bereitschaft zur Unterstützung sowie seine ausführlichen und konstruktiven Rückmeldungen waren maßgebliche Faktoren für das sehr positive Endergebnis dieses Projekts.

Ebenso möchten wir dem Geschäftsführer der Xinger Solutions GmbH, Georg Kreuzinger, unseren Dank aussprechen. Seine Motivation und sein Interesse an der gemeinsamen Umsetzung dieses Projekts sowie seine langjährige Projekterfahrung waren von großem Wert, insbesondere bei der konzeptionellen Planung und strategischen Ausrichtung der Arbeit.

Darüber hinaus danken wir unseren Freunden und Familien für ihre Unterstützung während der gesamten Entwicklungsphase. Durch ihre Ermutigung, ihr Verständnis und ihre regelmäßigen Rückmeldungen haben sie wesentlich zur erfolgreichen Fertigstellung dieser Diplomarbeit beigetragen.

1.5. Gendererklärung

Zur besseren Lesbarkeit der Diplomarbeit wurde ausschließlich die männliche Form verwendet. Da Begriffe wie „Benutzerinnen und Benutzer“ den Text unleserlich machen, wurde es schlicht auf „Benutzer“ gekürzt, dies soll jedoch keine Geschlechterdiskriminierung zum Ausdruck bringen.

2. Einleitung

2.1. Ausgangslage und Motivation der Arbeit

Literaturrecherche, ähnliche Projekte, ...

2.2. Spezifische Ausgangslage

Die zentrale Aufgabe dieser Diplomarbeit besteht in der Konzeption und Entwicklung einer mobilen Applikation, die sowohl auf iOS- als auch auf Android-Endgeräten lauffähig ist. Bereits zu Beginn des Projekts ergab sich eine grundlegende technische Fragestellung hinsichtlich der Wahl eines geeigneten Frameworks, das eine plattformübergreifende Entwicklung ermöglicht, gleichzeitig jedoch den Anforderungen an Performance, Wartbarkeit und Erweiterbarkeit gerecht wird. Diese Entscheidung stellte einen wesentlichen Einflussfaktor auf die gesamte weitere Projektumsetzung dar.

Für die inhaltliche Ausgestaltung der Applikation diente das bekannte Online-Spiel GeoGuessr als konzeptionelle Inspiration. Dieses Spielprinzip basiert auf der zufälligen Platzierung von Spieler an realen Orten, die anhand visueller Hinweise identifiziert werden müssen. Im Zuge erster Analysen zeigte sich jedoch, dass die globale Ausrichtung dieses Konzepts für viele Nutzer eine erhebliche Einstiegshürde darstellt. Insbesondere Personen ohne ausgeprägtes geografisches Fachwissen stoßen rasch an ihre Grenzen, was sowohl den Spielspaß als auch die langfristige Nutzung beeinträchtigen kann.

Aus dieser Beobachtung heraus entstand die Idee zur Entwicklung von WoSamma, einer Applikation, die das bewährte Spielprinzip aufgreift, dieses jedoch gezielt auf Österreich beschränkt. Durch die regionale Eingrenzung soll einerseits die Zugänglichkeit erhöht und andererseits ein spielerischer Lernmehrwert geschaffen werden. Gleichzeitig eröffnet dieses klar definierte Zielgebiet neue Möglichkeiten hinsichtlich der inhaltlichen Ausgestaltung, Nutzerbindung sowie wirtschaftlichen Verwertung der Anwendung.

Im Zuge der Projektplanung rückten neben der technischen Realisierung zunehmend auch wirtschaftliche und strukturelle Fragestellungen in den Fokus. Insbesondere stellte sich die Frage, wie eine solche Anwendung langfristig betrieben werden kann, ohne die Nutzererfahrung negativ zu beeinflussen, sowie wie die zugrunde liegende Systemarchitektur gestaltet sein muss, um auch bei steigenden Nutzerzahlen stabil und performant zu bleiben. Diese Überlegungen bilden die Grundlage für die im weiteren Verlauf der Arbeit behandelten spezifischen Forschungsfragen.

2.3. Spezifische Forschungsfrage

2.3.1. Spezifische Forschungsfrage - Jan Tiefenbacher

Welche Monetarisierungsmodelle sind für diese App am effektivsten, um eine nachhaltige Einnahmengenerierung zu ermöglichen und gleichzeitig eine ausgewogene Balance zwischen Wirtschaftlichkeit und Nutzerzufriedenheit zu gewährleisten?

2.3.2. Spezifische Forschungsfrage - Laurenz Pichler

Welche technischen und organisatorischen Skalierungsstrategien ermöglichen es, eine bestehende App-Infrastruktur kurzfristig und langfristig an stark wachsende Nutzerzahlen anzupassen?

3. Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen vorgestellt, die für das Verständnis der entwickelten Anwendung erforderlich sind. Dabei werden zentrale Begriffe, Konzepte und technische Zusammenhänge erläutert, die als Basis für die nachfolgenden Kapitel und die technische Umsetzung dienen.

3.1. Plattformwahl

Die Wahl der Zielplattform ist ein entscheidender Schritt bei der Entwicklung mobiler Anwendungen. Sie beeinflusst maßgeblich die technische Umsetzung, das Design, die Verfügbarkeit und die Monetarisierungsmöglichkeiten der App. Im folgenden Abschnitt werden die wichtigsten mobilen Betriebssysteme vorgestellt und die Unterschiede zwischen nativen und Cross-Platform-Apps erläutert. Abschließend wird die getroffene Entscheidung für die vorliegende Arbeit begründet.

iOS
iOS ist das mobile Betriebssystem von Apple, welches auf Geräten wie den Iphones oder Ipads verwendet wird. Es ist für Touch-Bedienung konzipiert und bekannt für hohe Sicherheitsstandards und eine intuitive Benutzeroberfläche. Technisch basiert iOS auf einem Unix-ähnlichen System-Kernel namens Darwin, was eine solide, stabile Basis für moderne mobile Anwendungen schafft. Apple verbindet Hard und Software sehr eng miteinander und kontrolliert die Plattformen stark, daher können Updates, Sicherheitsmechanismen und Apple-Dienste über alle Geräte sehr einheitlich bereitgestellt werden. Um eine entwickelte Applikation für iOS Geräte verfügbar zu machen, muss diese auf dem Apple App Store veröffentlicht werden, wobei durch strenge Richtlinien und Prüfprozesse sichergestellt wird, dass nur qualitativ hochwertige und sichere Apps zugelassen werden. Außerdem braucht man eine Apple Developer Lizenz, um Apps im App Store veröffentlichen zu können.[1] [2] [3]

Android
Android ist ein von Google entwickeltes, Linux-basiertes und mobiles Betriebssystem, das auf Smartphones und Tablets läuft und als Plattform alle System- und Benutzerkomponenten umfasst, also das Linux-Kernel-Betriebssystem, die grafische Oberfläche und die verfügbaren Apps. Android wurde unter der Apache-Open-Source-Lizenz veröffentlicht, dies erlaubt Hertsellern und Entwicklern, die Software zu individuellen Varianten zu modifizieren. Obwohl die Basis offen ist, enthalten die meisten Geräte zusätzliche Basis-Programme wie Google-Apps, die vorinstalliert sind. Android zählt daher zu den am meisten verbreiteten Betriebssystemen weltweit. Um eine Applikation für Android-Geräte verfügbar zu machen, wird diese in der Regel über den Google Play Store vertrieben. Auch hier gibt es Richtlinien und Prüfverfahren, welche sicherstellen, dass nur funktionierende und sichere Apps veröffentlicht werden. Zusätzlich benötigt man ein Google Developer-Konto, um Anwendungen im Play Store bereitstellen zu können. [4] [5]

Plattformspezifische Unterschiede zwischen iOS und Android
iOS- und Android-Systeme unterscheiden sich in vielen Bereichen, darunter in ihrer Systemarchitektur, Designrichtlinien und Gerätevielfalt. Während iOS auf eine begrenzte Anzahl an

Geräten optimiert ist, existiert im Android-Bereich eine große Vielfalt an Bildschirmgrößen und Hardwarekonfigurationen. Diese Unterschiede erhöhen den Entwicklung- und Wartungsaufwand bei nativen Anwendungen, also Anwendung, welche speziell für ein bestimmtes Betriebssystem entwickelt wurden.

Native Apps – Technische Grundlagen
Bei der nativen App-Entwicklung handelt es sich um die Entwicklung von Anwendungen, die speziell für ein bestimmtes Betriebssystem programmiert werden. Für iOS-Anwendungen wird beispielsweise die Programmiersprache Swift verwendet, während Android-Applikationen typischerweise in Kotlin entwickelt werden. Die Anwendung wird direkt für das jeweilige Betriebssystem entwickelt und ist nicht mit anderen Systemen kompatibel.

Ein wesentliches Merkmal nativer Anwendungen ist der direkte Zugriff auf systemnahe Funktionen und Hardware-Komponenten des Endgeräts, wie Kamera, GPS oder Sensoren. Native Apps können zudem systeminterne Funktionen wie Push-Benachrichtigungen nutzen. Durch die enge Integration mit dem Betriebssystem ist eine optimale Nutzung von Ressourcen wie Arbeitsspeicher und Hardware möglich, was zu einer guten Performance und hoher Usability führt.

Allerdings erfordert dieser Ansatz für jedes Betriebssystem eine eigene Implementierung, wodurch separate Versionen für iOS und Android entwickelt werden müssen. Dies erhöht den Entwicklungsaufwand deutlich. [6] [7] [8]

Native Apps – Design und Monetarisierung
Native Apps orientieren sich stark an den Design- und Interaktionsrichtlinien des jeweiligen Betriebssystems. Dadurch fügen sie sich nahtlos in die Benutzeroberfläche von iOS oder Android ein und bieten eine konsistente User Experience. Die hohe Usability entsteht durch die Verwendung systemeigener UI-Elemente und Interaktionsmuster, die den Nutzern bereits vertraut sind.

Die Bereitstellung nativer Apps erfolgt über die jeweiligen App Stores der Betriebssysteme. Dadurch sind sie direkt an das Ökosystem der Plattform gebunden und können die dort vorgesehenen Mechanismen zur Verteilung und Nutzung verwenden. [6] [7] [8]

Cross-Platform Apps – Technische Grundlagen
Bei der Cross-Platform-App-Entwicklung wird eine gemeinsame Codebasis verwendet, um Anwendungen für mehrere Betriebssysteme wie iOS und Android zu realisieren. Mithilfe spezieller Frameworks wie Flutter, React Native oder Xamarin wird dieser Code für die jeweiligen Plattformen umgesetzt. Ziel dieses Ansatzes ist es, große Teile des Codes plattformübergreifend wiederzuverwenden.

Cross-Platform-Frameworks abstrahieren die zugrunde liegenden Betriebssysteme und stellen einheitliche Schnittstellen zur Verfügung. Dadurch ist es möglich, eine Anwendung zu entwickeln, ohne für jedes Betriebssystem eine vollständig eigene Implementierung zu erstellen. Die Interaktion mit gerätespezifischen Funktionen erfolgt dabei häufig über zusätzliche Abstraktionsschichten oder spezielle Erweiterungen.

Durch die gemeinsame Codebasis verringert sich der Entwicklungsaufwand, und Änderungen müssen nur einmal durchgeführt werden, um sie auf allen unterstützten Plattformen verfügbar

zu machen. Dies erleichtert zudem die Wartung der Anwendung. [6] [7] [8]

Cross-Platform Apps – Design und Monetarisierung
Cross-Platform-Apps können viele native Funktionen nutzen und sich in vielerlei Hinsicht wie native Anwendungen anfühlen, auch wenn sie nicht vollständig plattformspezifisch entwickelt wurden. Das Design ist dabei weniger strikt an die Richtlinien eines einzelnen Betriebssystems gebunden, da eine einheitliche Umsetzung für mehrere Plattformen angestrebt wird.

Durch die schnellere Markteinführung und den geringeren Entwicklungsaufwand eignet sich dieser Ansatz besonders für Anwendungen, die gleichzeitig auf mehreren Plattformen verfügbar sein sollen. [6] [7] [8]

Cross-Platform-Lösung
Bevor mit der eigentlichen Entwicklung der App begonnen werden konnte, musste zunächst eine Entscheidung bezüglich der Zielplattform getroffen werden. Aufgrund der definierten Zielgruppe sowie der angestrebten Reichweite fiel die Wahl auf eine Cross-Platform-Lösung, um sowohl iOS- als auch Android-Nutzer zu erreichen.

Einsatz von React Native (Kurzüberblick)
Für die Umsetzung der App wurde React Native als Cross-Platform-Framework gewählt, um eine gemeinsame Codebasis für iOS und Android zu ermöglichen. (Technische Details folgen in Abschnitt: 3.0.4.1.1. React Native Framework)

3.2. UI/UX-Design & Nutzererlebnis

Im folgenden Abschnitt wird auf die Informationsarchitektur, Designkonzepte und Navigation eingegangen. Dabei wird erläutert, wie Inhalte sinnvoll strukturiert, gestaltet und geführt werden, um eine klare Orientierung und eine intuitive Nutzung digitaler Anwendungen zu ermöglichen.

3.2.1. Trends, Designkonzepte & Weiterentwicklungen im App-Design

Nun werden zentrale Trends und Weiterentwicklungen im App-Design vorgestellt, die 2025 und die Jahre danach maßgeblich beeinflussen und wie Nutzer digitale Produkte wahrnehmen und verwenden.

Modernes App Design
Im Jahr 2025 hat sich Design im digitalen Kontext von einer rein visuellen Disziplin zu einem zentralen strategischen Faktor entwickelt. In einem stark gesättigten Markt mit einer Vielzahl an Apps und digitalen Services entscheidet nicht mehr allein die Funktionalität über den Erfolg eines Produkts, sondern vor allem die Qualität der User Experience. Nutzer erwarten intuitive, schnelle und personalisierte Anwendungen, die sich nahtlos in ihren Alltag integrieren. Design beeinflusst dabei direkt Nutzerbindung, Verweildauer und Akzeptanz digitaler Produkte.

Dark und Light Mode
Der Dark Mode hat sich in der modernen Web- und App-Entwicklung längst als Standard etabliert und ist nicht mehr nur eine optionale Designentscheidung. Während früher der Light Mode als Marktstandard galt und der Dark Mode lediglich als Zusatzfunktion angeboten wurde, hat sich dieses Verhältnis in den letzten Jahren komplett gewandelt. Heute setzen die meisten mobilen Anwendungen standardmäßig auf den Dark Mode und bieten, wenn überhaupt, einen optionalen Light Mode an. Neben ästhetischen Aspekten überzeugt der Dark Mode vor allem durch ergonomische Vorteile wie eine reduzierte Augenbelastung, insbesondere in dunklen Umgebungen, sowie potenzielle Energieeinsparungen auf OLED- und AMOLED-Displays. Moderne Designs berücksichtigen daher unterschiedliche Lichtverhältnisse und Nutzungskontexte und passen Kontraste sowie Farbschemata dynamisch an, um sowohl Nutzbarkeit als auch Zugänglichkeit zu optimieren. Dennoch bleibt der Light Mode in hellen Umgebungen oder für bestimmte Nutzergruppen weiterhin relevant, weshalb eine flexible Umschaltmöglichkeit zwischen beiden Modi als bewährte Praxis gilt.[9] [10]

Responsive & Adaptive Design
Responsive und Adaptive Design beschreibt zwei zentrale Ansätze moderner Web und App Gestaltung, die als Reaktion auf die starke Diversifizierung internetfähiger Endgeräte entstanden sind. Während vor dem mobilen Web weitgehend homogene Bildschirmgrößen dominierten, müssen Anwendungen heute auf Displaybreiten von etwa 320 Pixel bis über 4.000 Pixel sowie unterschiedliche Eingabemethoden und Auflösungen reagieren. Responsive Design verfolgt dabei einen flexiblen, fließenden Ansatz, bei dem sich ein einziges Layout mithilfe relativer Einheiten dynamisch an den verfügbaren Bildschirmplatz anpasst, um eine konsistente User Experience auf allen Geräten zu gewährleisten. Adaptive Design hingegen arbeitet mit mehreren vordefinierten, eher starren Layouts, die abhängig von Geräteeigenschaften wie Bildschirmgröße oder Ausrichtung geladen werden und häufig feste Pixelwerte verwenden. Beide Ansätze zielen darauf ab, Usability, Performance und Designqualität zu optimieren, unterscheiden sich jedoch in ihrer Philosophie. Während Responsive Design das Verhalten der Inhalte definiert, legt Adaptive Design das konkrete Darstellungsergebnis für bestimmte Gerätekategorien fest. In der Praxis werden die Vorteile beider Konzepte oft kombiniert, um sowohl Flexibilität als auch gezielte Optimierung für ausgewählte Endgeräte zu erreichen. [11] [12]

Accessibility & Barrierefreiheit
Accessibility bzw. Barrierefreiheit beschreibt das Ziel, digitale Produkte so zu gestalten und umzusetzen, dass sie von möglichst allen Menschen gleichwertig genutzt werden können. Das betrifft nicht nur Personen mit dauerhaften Einschränkungen wie Seh- oder Hörbehinderungen, motorischen oder kognitiven Beeinträchtigungen, sondern auch situative Einschränkungen, etwa wenn jemand kurzfristig eine verletzte Hand hat oder bei starker Sonne kaum etwas am Display erkennt. Barrierefreiheit ist damit kein „Extra-Feature“, sondern ein Qualitätsmerkmal guter User Interfaces. Wenn eine App klar strukturiert, gut bedienbar und verständlich ist, profitieren am Ende alle Nutzer. [13] [14]

Warum Barrierefreiheit wichtig ist
Barrierefreiheit hat mehrere Ebenen an Relevanz. Aus ethischer Sicht geht es um digitale Teilhabe, Apps und Websites sind heute zentrale Zugänge zu Information, Kommunikation und

Services, weshalb Ausschlüsse durch schlechtes Design reale Konsequenzen haben. Zusätzlich spielt eine rechtliche Dimension hinein, da Accessibility-Anforderungen in vielen Ländern und Branchen stärker reguliert werden. Und auch wirtschaftlich ist das Thema relevant. Barrierefreie Produkte verbessern oft die Markenwahrnehmung, erhöhen die Nutzerbindung und erschließen zusätzliche Zielgruppen, statt potenzielle User einfach auszulassen.[13] [14]

Struktur, Hierarchie und Fokusführung
Ein Kernpunkt barrierefreier Gestaltung ist eine klare Informationshierarchie. Nutzer müssen schnell erkennen können, wo sie sind und was die wichtigsten Aktionen sind. Dabei ist nicht nur das visuelle Layout relevant, sondern auch die Reihenfolge, in der Inhalte technisch angeordnet sind. Screenreader lesen Inhalte typischerweise in einer top-down Reihenfolge aus dem Markup. Deshalb ist die Zusammenarbeit zwischen Design und Development wichtig, damit visuelle Hierarchie, DOM-Reihenfolge und Fokuslogik zusammenpassen. Elemente sollen in einer logischen Reihenfolge erreichbar sein und Gruppierungen sollen verständlich sein, damit Nutzer sich schnell orientieren können. [13] [14]

Kontrast, Farbe und Typografie
Damit Inhalte für möglichst viele Menschen wahrnehmbar sind, spielen Kontrast und Lesbarkeit eine zentrale Rolle. Ausreichende Kontraste zwischen Text und Hintergrund helfen insbesondere bei Sehbehinderungen, aber auch in Alltagssituationen wie starkem Umgebungslicht. Wichtig ist außerdem, Informationen nicht ausschließlich über Farbe zu kommunizieren, beispielsweise einen Fehler nur Rot zu markieren, sondern zusätzliche Hinweise wie Text, Icons oder Umrandungen zu verwenden. Typografie und Layout sollten so angelegt sein, dass größere Schriftgrößen und Zoom nicht zu überlappenden oder abgeschnittenen Elementen führen. Flexible Layouts, ausreichende Abstände und skalierbare Schriftgrößen sind hier zentrale Bausteine.[13] [14]

Touch Targets und Eingabemethoden
Gerade auf mobilen Geräten ist die Größe von Interaktionsflächen entscheidend. Kleine Icons ohne ausreichende Abstände führen schnell zu Fehlbedienungen, besonders bei motorischen Einschränkungen. Deshalb sollten Buttons, Icons und interaktive Elemente genügend große Touch- bzw. Pointer-Flächen haben und mit ausreichendem Abstand zueinander platziert werden.[13] [14]

Labels und Alt-Text
Ein weiterer zentraler Baustein ist eine klare und aussagekräftige Textgestaltung innerhalb der Anwendung. Dazu zählen sichtbare Beschriftungen wie Buttontexte sowie kurze beschreibende Texte für Symbole und grafische Elemente. Diese Inhalte sollten präzise, eindeutig und handlungsorientiert formuliert sein, um eine schnelle Orientierung und effiziente Nutzung zu ermöglichen.

Auch bei Bildern ist eine bewusste Textzuordnung wichtig, sofern sie relevante Informationen vermitteln. Der Text sollte sich dabei auf das Wesentliche beschränken und den inhaltlichen Zweck des Bildes erklären.[13] [14]

Testing und Umsetzung
Ein qualitativ hochwertiges User Interface entsteht durch kontinuierliches Testen und an-

schließende Anpassung der Anwendung. Regelmäßige Usability-Tests sind notwendig, um zu überprüfen, ob Struktur, Navigation und Interaktionsabläufe den Erwartungen der Nutzer entsprechen. Dabei zeigen Tests häufig Schwachstellen oder Fehlannahmen auf, die im Entwicklungsprozess nicht offensichtlich sind.

Auf Basis der Testergebnisse muss das Design iterativ angepasst und optimiert werden. Durch diesen wiederkehrenden Prozess aus Testen, Auswerten und Anpassen kann sichergestellt werden, dass die Anwendung langfristig benutzerfreundlich, verständlich und effektiv bleibt.

3.2.2. Informationsarchitektur & Navigation

Die besten Inhalte entfalten nur dann Wirkung, wenn Nutzer sie schnell finden und einordnen können. Genau hier setzt die Informationsarchitektur an. Sie beschreibt die Strukturierung, Gliederung und Verknüpfung von Informationen in digitalen Anwendungen. Ziel ist es, eine logische Hierarchie zu schaffen, die Inhalte verständlich organisiert und dadurch die Auffindbarkeit sowie die Orientierung verbessert. Informationsarchitektur wirkt meist „unsichtbar“, beeinflusst aber unmittelbar, ob eine Website oder eine App als klar und intuitiv oder als unübersichtlich wahrgenommen wird. [15] [16]

Informationsarchitektur ist das konzeptionelle „Gerüst“ einer Website oder Anwendung. Sie umfasst nicht nur die Anordnung von Inhalten innerhalb eines Hauptmenüs, sondern auch das „Wie und Wo“. Welche Inhalte und Funktionen existieren, wie sind sie gebündelt, wie hängen sie zusammen und an welchen Stellen werden sie angeboten. Damit betrifft Informationsarchitektur auch die Struktur einzelner Seiten, bezüglich der Platzierung von Modulen wie Menüs, Suchfunktionen, Newsfeeds oder Profilbereichen. [15] [16]

Auf Basis etablierter Informationsarchitektur-Modelle lässt sie sich in vier zentrale Komponenten gliedern:[15] [16]

- Organisationssysteme: Definieren, wie Inhalte gruppiert und kategorisiert werden (z. B. thematisch, alphabetisch, nach Nutzerrollen).
- Navigationssysteme: Bestimmen, wie Nutzer durch die Informationsstruktur geführt werden (z. B. Menüs, Suchfunktionen).
- Suchsysteme: Ermöglichen das gezielte Auffinden von Inhalten über Suchfunktionen und Filter.
- Kennzeichnungssysteme: Sorgen für klare Beschriftungen und Metadaten, die Inhalte verständlich machen.

Die Informationsarchitektur wird idealerweise bereits in frühen Projektphasen definiert, sie sorgt für eine bessere User Experience, indem sie Klarheit schafft und Suchzeiten minimiert. Sie sorgt außerdem für mehr energetische Effizienz, da der User weniger Klicks braucht um ans Ziel zu kommen.[15] [16]

Ein gutes Informationsarchitektur-Design achtet auf folgende Prinzipien:[15][16]

- begrenzte Auswahl pro Ebenen
- angemessene Offenlegung von Informationen
- Mehrfachklassifikation: mehrere Wege führen zum Ziel
- Skalierbarkeit: die Struktur sollte zukünftiges Wachstum und neue Inhalte berücksichtigen.

3.2.3. Wireframes, Mockups & Prototypen

Diese Design-Artefakte sind zentrale Bestandteile des modernen Designprozesses für digitale Produkte wie Websites, Webanwendungen oder mobile Apps. Sie dienen dazu, Ideen frühzeitig zu strukturieren, zu visualisieren und zu überprüfen, bevor zeit- und kostenintensive Entwicklungsarbeiten beginnen. Obwohl die Begriffe im Alltag häufig synonym verwendet werden, erfüllen Wireframes und Prototypen unterschiedliche Aufgaben und besitzen jeweils eigene Eigenschaften und Zielsetzungen. [17][18][19]

Wireframes: Struktur und Funktion als Grundlage

Ein Wireframe ist eine vereinfachte, meist statische Darstellung eines digitalen Produkts. Er bildet das grundlegende Gerüst einer Anwendung ab und konzentriert sich auf Struktur, Layout und Informationsarchitektur. Im Vordergrund steht die Frage, welche Inhalte wo platziert werden und wie Nutzer durch das System geführt werden. Visuelle Details wie Farben, Typografie oder Bilder spielen dabei eine untergeordnete Rolle oder fehlen vollständig. Häufig werden Wireframes in Graustufen mit einfachen Boxen, Linien und Platzhaltern umgesetzt.

Wireframes werden vor allem in frühen Phasen des Designprozesses eingesetzt. Sie ermöglichen es, Konzepte schnell zu erstellen, zu diskutieren und zu verändern. Dadurch eignen sie sich besonders gut, um erste Ideen abzustimmen und grundlegende Designentscheidungen zu treffen. Ein weiterer Vorteil liegt in der Kosten- und Zeiteffizienz, da Anpassungen ohne großen Aufwand vorgenommen werden können. Gleichzeitig liegt hier auch eine Einschränkung. Wireframes haben kaum Interaktivität und keine realistischen Inhalte, daher sind sie nur bedingt geeignet, um Benutzerfreundlichkeit oder Nutzerinteraktionen zu testen.[17][18][19]

Prototypen: Interaktion und Realitätsnähe

Prototypen bauen auf Wireframes auf und stellen eine weiterentwickelte, interaktive Version des Produkts dar. Sie reichen von einfachen Low-Fidelity-Prototypen (Prototypen mit einer sehr einfachen, groben Darstellung, die sich auf Struktur, Layout und grundlegende Abläufe konzentriert, ohne visuelle Details oder echte Interaktionen einzubauen), bis hin zu High-Fidelity-Prototypen (stark ausgearbeitete Prototypen, sie zeigen Farben, Typografien, echte Inhalte und oft Interaktionen), die dem späteren Endprodukt visuell und funktional sehr nahekommen. Prototypen enthalten reale Inhalte, UI-Elemente, Animationen und definierte Interaktionen. Nutzer können klicken, navigieren und Abläufe realistisch nachvollziehen.[17][18][19][20]

Der Hauptzweck von Prototypen liegt in der Validierung von Designentscheidungen. Durch Nutzertests lassen sich Missverständnisse oder Schwächen im Benutzerfluss frühzeitig erkennen. Dadurch kann wertvolles Feedback gesammelt werden, bevor mit der technischen Umsetzung begonnen wird. Zwar ist die Erstellung von Prototypen aufwendiger als die von Wireframes, langfristig helfen sie jedoch, Fehlentwicklungen und teure Nachbesserungen zu vermeiden.[17][18][19][20]

Mockups: Visueller Designprozess
Neben Wireframes und Prototypen werden häufig auch Mockups verwendet. Mockups sind statische, visuell ausgearbeitete Darstellungen eines Produkts. Sie zeigen Farben, Typografie und das visuelle Erscheinungsbild sehr genau, bieten jedoch keine Interaktivität. Während Wireframes die Struktur festlegen und Prototypen die Nutzung simulieren, dienen Mockups vor allem der Beurteilung des visuellen Designs. Oft wird aber auch auf Mockups verzichtet wenn der Prototyp bereits stark visuell ausgearbeitet ist (High-Fidelity-Prototypen).[17][18][19][20]

Zusammenspiel im Designprozess
Die verschiedenen Design-Artefakte sind keine konkurrierenden Phasen, sondern ergänzen sich und bauen aufeinander auf. In der Regel beginnt der Designprozess mit Wireframes, die als Fundament des Designs dienen. Danach werden Mockups angefertigt um das visuelle Erscheinungsbild zu evaluieren und perfektionieren. Sobald Struktur und Funktionalität festgelegt sind, werden diese Entwürfe in Prototypen überführt, um das Nutzungserlebnis realistisch darzustellen und zu testen.[17][18][19]

3.2.4. Visuelles Design

Das Visuelles Design bezeichnet die Gestaltung der Inhalte mit dem Ziel, Applikationen und digitae Inhalte visuell ansprechend und zugleich funktional zu gestalten. Es bildet eine zentrale Schnittstelle zwischen Gestaltung und Nutzererfahrung, da es nicht nur das äußere Erscheinungsbild eines Produkts prägt, sondern auch im Wesentlichen beeinflusst, wie Inhalte wahrgenommen, verstanden und genutzt werden. In der digitalen Welt ist visuelles Design daher ein wichtiger Bestandteil von Webanwendungen, Software, Marketingplattformen und interaktiven Medien.[21][22]

Bestandteile des visuellen Designs
Im Kern umfasst visuelles Design alle sichtbaren Gestaltungskomponenten eines digitalen Produkts. Dazu zählen Bilder, grafische Elemente, Farben, Typografie, Layoutstrukturen sowie der gezielte Einsatz von Weißraum. Diese Elemente wirken nicht isoliert, sondern entfalten ihre Wirkung im Zusammenspiel. Ein konsistentes und durchdachtes visuelles Design trägt zur Stärkung der Markenidentität bei, erhöht die Wiedererkennbarkeit und unterstützt eine klare Kommunikation von Inhalten.[21][22]

Zentrale Gestaltungselemente
Besondere Bedeutung kommt den einzelnen Gestaltungselementen zu. Bilder und grafische

Elemente dienen nicht nur der Dekoration, sondern übernehmen eine informative und emotionale Funktion. Sie können komplexe Inhalte vereinfachen, Aufmerksamkeit lenken und eine Verbindung zum Nutzer herstellen. Typografie beeinflusst maßgeblich die Lesbarkeit und visuelle Hierarchie. Schriftart, -größe und -gewicht bestimmen, welche Inhalte priorisiert wahrgenommen werden. Die Farbgestaltung wirkt stark auf die emotionale Ebene und kann Stimmungen erzeugen, Kontraste verstärken sowie Orientierung bieten.[21][22]

3.3. Monetarisierung von mobilen Apps

In diesem Kapitel wird der globale App-Markt und die verschiedenen Strategien zur Monetarisierung mobiler Apps vorgestellt.

3.3.1. Grundlagen & Marktüberblick

Um die wirtschaftliche und technologische Bedeutung mobiler Anwendungen einordnen zu können, ist ein grundlegender Überblick über den App-Markt erforderlich. Der folgende Abschnitt beleuchtet die Entwicklung des mobilen App-Marktes, zentrale App-Ökosysteme sowie aktuelles Nutzerverhalten und relevante Marktstatistiken.

Entwicklung des mobilen App-Marktes
Der Markt für Apps ist in den letzten Jahren stark gewachsen und es wird prognostiziert, dass dieser Trend auch in Zukunft anhalten wird. Laut Market Research Future (MRFR) wurde das Marktvolumen im Jahr 2024 auf rund 94,4 Milliarden USD geschätzt und soll von 116,87 Milliarden USD im Jahr 2025 auf etwa 988,5 Milliarden USD bis 2035 anwachsen. Wesentlich dazu tragen bei, dass E-Commerce immer weiter ausgebaut wird und auch küstliche Intelligenzen immer mehr in Mobile Applikationen integriert werden.[23]

App-Ökosysteme
Mobile Anwendungen werden in der Regel über zentrale App-Stores vertrieben, die als geschlossene Ökosysteme fungieren und sowohl die Distribution als auch die Monetarisierung von Apps steuern. Die beiden dominierenden Distributionsplattformen im globalen Markt für mobile App-Entwicklung sind der Apple App Store und der Google Play Store. Laut der Marktanalyse von Market Research Future entfällt ein erheblicher Anteil des weltweiten Umsatzes auf das iOS-Ökosystem. Bereits im Jahr 2021 machte iOS mehr als 62,88 % des globalen Umsatzes im Markt für mobile App-Entwicklung aus. Dies ist vor allem auf die höhere Zahlungsbereitschaft der Nutzer im Apple-Ökosystem sowie auf eine stärkere Verbreitung von Premium-Apps und In-App-Käufen zurückzuführen. Der Google Play Store hingegen verzeichnet zwar höhere Downloadzahlen, insbesondere durch die große Verbreitung von Android-Geräten, erzielt jedoch im Vergleich geringere Umsätze pro Nutzer. [23]

Nutzerverhalten & Marktstatistiken
Marktanalysen zeigen, dass Nutzer zunehmend bereit sind, für digitale Inhalte und Zusatzfunktionen innerhalb mobiler Anwendungen zu bezahlen. Besonders deutlich wird dieses

Verhalten im Bereich mobiler Spiele sowie bei abonnementbasierten Anwendungen. Der Bericht hebt hervor, dass ein wesentlicher Teil der Umsätze aus In-App-Käufen, Premium-Anwendungen und mobilen Games stammt. Darüber hinaus treibt die weltweit steigende Smartphone-Durchdringung das Nutzerwachstum kontinuierlich an. Bis 2025 wird erwartet, dass in entwickelten Regionen über 80 % der Bevölkerung ein Smartphone besitzen, was die Nachfrage nach mobilen Anwendungen weiter erhöht.[23]

3.3.2. Werbung als Einnahmequelle

Im kommenden Abschnitt wird Werbung als zentrale Einnahmequelle mobiler Anwendungen betrachtet. Dabei werden gängige Werbeformate, deren Einfluss auf die Nutzererfahrung sowie die Rolle von Werbenetzwerken und Vergütungsmodellen im App-Markt erläutert.

Arten von Werbung in mobilen Apps
In-App-Werbung umfasst Werbeanzeigen, die direkt innerhalb einer mobilen Anwendung ausgespielt werden. Ziel ist es, Apps kostenlos oder günstiger anbieten zu können und gleichzeitig Einnahmen zu generieren. Dabei existieren verschiedene Werbeformate, die sich in Platzierung, Nutzerinteraktion und Einfluss auf die User Experience unterscheiden. Je nach App-Typ (z.B. Gaming, Lifestyle, News) und Zielgruppe werden unterschiedliche Formate bevorzugt eingesetzt.[24][25]

Banner-Werbung
Banner sind kleine, statische oder animierte Anzeigen, die meist am oberen oder unteren Bildschirmrand eingeblendet werden. Sie gelten als vergleichsweise unaufdringlich, bieten jedoch nur begrenzte Interaktionsmöglichkeiten. Wichtig ist eine Platzierung, die keine Inhalte verdeckt und keine unbeabsichtigten Klicks durch Scrollen oder Wischen provoziert, da dies den User negativ auffällt.[24][25]

Interstitial-Werbung
Interstitials sind Vollbildanzeigen, die zwischen App-Inhalten erscheinen, beispielsweise beim Wechsel zwischen Seiten oder nach dem Abschließen eines Levels in Spielen. Sie erzeugen hohe Aufmerksamkeit, können aber die Nutzung stark unterbrechen und werden deshalb schnell als störend wahrgenommen, wenn sie zu häufig oder zu ungünstigen Zeitpunkten eingeblendet werden.[24][25]

Videoanzeigen
Videoanzeigen eignen sich besonders gut zur emotionalen und narrativen Vermittlung von Inhalten, da sie Bewegtbild, Ton und zeitliche Dramaturgie kombinieren. Man unterscheidet dabei unter anderem zwischen In-Stream- und Outstream-Formaten. In-Stream-Videoanzeigen werden vor, während oder nach einem redaktionellen oder spielbezogenen Videoinhalt abgespielt, während Outstream-Formate unabhängig von einem klassischen Video-Kontext beispielsweise innerhalb von Feeds oder zwischen textbasierten Inhalten erscheinen.

Insbesondere in Gaming-Apps haben sich Videoanzeigen als etabliertes Werbeformat durchgesetzt, vor allem in Form von sogenannten Rewarded Videos, bei denen Nutzer freiwillig eine Werbeanzeige ansehen und dafür eine spielinterne Belohnung erhalten. Videoanzeigen erzielen im Vergleich zu klassischen Banneranzeigen in der Regel höhere Einnahmen, da

sie eine stärkere Aufmerksamkeit erzeugen, höhere Interaktionsraten aufweisen und von Werbetreibenden entsprechend höher vergütet werden. [24][25]

Native-Werbung
Native Ads sind Anzeigen, die optisch und inhaltlich möglichst nahtlos in das Design und den Kontext der App integriert werden (z.,B. im Feed von Social Media oder in News-Apps). Dadurch wirken sie weniger wie klassische Werbung und werden häufig besser akzeptiert. Der Vorteil liegt in der geringeren Störung des Nutzungserlebnisses bei gleichzeitig guter Sichtbarkeit.[24][25]

Playable-Werbung
Playable Ads sind interaktive Anzeigen, bei denen Nutzer ein Produkt kurz testen können, bevor sie es herunterladen. Diese Form ist besonders effektiv im Gaming-Bereich, da sie Interaktion statt passivem Konsum ermöglicht und oft eine hohe Conversion-Rate erzielt.[24][25]

Display-Anzeigen
Zusätzlich werden klassische Display-Anzeigen genutzt, die aus Text- oder Bildinhalten bestehen. Gestaltung, Form und Größe sind flexibel, wodurch sie in vielen App-Typen einsetzbar sind. Wie bei Bannern hängt die Effektivität stark von Platzierung und Kontext ab.[24][25]

Werbenetzwerke
Werbenetzwerke spielen eine zentrale Rolle bei der Monetarisierung mobiler Anwendungen, da sie App-Entwickler (Publisher) mit Werbetreibenden verbinden. Technisch erfolgt die Einbindung von Werbung in der Regel über sogenannte Software Development Kits (SDKs), die in die App integriert werden. Diese SDKs übernehmen wesentliche Funktionen wie das Ausspielen von Anzeigen, die Durchführung von Echtzeitauktionen, das Tracking von Impressionen und Klicks sowie die Abrechnung der erzielten Einnahmen. Moderne Werbenetzwerke arbeiten dabei häufig programmatisch und nutzen Real-Time-Bidding-Verfahren (RTB), um für jeden Anzeigenplatz automatisch das wirtschaftlich beste Werbemittel auszuwählen.[26]

Ein Werbenetzwerk agiert dabei als Vermittler zwischen Angebotsseite (App mit verfügbarem Anzeigeninventar) und Nachfrageseite (Werbetreibende). Je nach Netzwerk und technischer Architektur sind Werbenetzwerke entweder direkt mit Demand-Side-Plattformen (DSPs) verbunden oder in größere Ökosysteme aus Supply-Side-Plattformen (SSPs) und Ad Exchanges eingebettet. Ziel ist es, möglichst relevante Werbung auszuspielen und gleichzeitig die Einnahmen für die App-Betreiber zu maximieren.[26]

Zu den häufig genutzten Werbenetzwerken im Bereich der mobilen App-Entwicklung zählen:

- **Google AdMob**
  Google AdMob ist eines der weltweit am weitesten verbreiteten mobilen Werbenetzwerke und Teil des Google-Werbeökosystems. Es ermöglicht Entwicklern einen einfachen Einstieg in die App-Monetarisierung und unterstützt zahlreiche Anzeigenformate wie Banner, Interstitials, Native Ads und Videoanzeigen. AdMob profitiert von der großen Reichweite und der hohen Nachfrage seitens der Werbetreibenden, wodurch auch kleinere Apps relativ früh Einnahmen erzielen können.[26]
- **Smaato**
  Smaato ist eine mobile Werbeplattform mit starkem Fokus auf programmatische Wer-

bung. Das Netzwerk agiert primär als Supply-Side-Plattform und verbindet App-Anbieter mit einer Vielzahl von Demand-Quellen. Smaato wird häufig in Kombination mit Mediation-Plattformen eingesetzt, um die Auslastung des Anzeigeninventars (Fill Rate) zu erhöhen und Wettbewerb zwischen Werbetreibenden zu schaffen.[26]

- **Meta Audience Network**
  Das Meta Audience Network (ehemals Facebook Audience Network) nutzt Daten aus dem Meta-Ökosystem, um zielgerichtete Werbung innerhalb mobiler Apps auszuspielen. Besonders im Bereich personalisierter Werbung bietet dieses Netzwerk Vorteile, da Anzeigen anhand von Interessen, Nutzungsverhalten und demografischen Merkmalen optimiert werden können. Es wird häufig in Social-, Gaming- und Lifestyle-Apps eingesetzt.[26]
- **AppLovin**
  AppLovin ist ein stark auf App- und insbesondere Game-Monetarisierung spezialisiertes Werbenetzwerk. Es unterstützt Echtzeit-Bidding und wird oft über Mediation-Plattformen wie AppLovin MAX integriert. Durch Live-Auktionen konkurrieren Werbetreibende um Anzeigenplätze, was potenziell zu höheren Einnahmen pro Impression führen kann. AppLovin ist vor allem bei Apps mit hoher Nutzerinteraktion und größeren Nutzerzahlen verbreitet.[26]

CPI im Kontext von Werbenetzwerken
Neben klassischen Vergütungsmodellen wie CPM (Cost Per Mille) oder CPC (Cost Per Click) spielt bei vielen Werbenetzwerken auch das CPI-Modell (Cost Per Install) eine wichtige Rolle. Beim CPI erhält der App-Entwickler eine Vergütung, wenn ein Nutzer über eine Anzeige eine beworbene App installiert. Dieses Modell wird besonders häufig im Gaming-Bereich eingesetzt, da dort App-Installationen ein zentrales Kampagnenziel darstellen. CPI bietet zwar oft höhere Einzelvergütungen als impressionbasierte Modelle, ist jedoch stärker von der Conversion-Rate abhängig und daher weniger planbar. In der Praxis kombinieren viele Entwickler CPI mit anderen Modellen oder binden mehrere Werbenetzwerke über Mediation ein, um Ertrag und Stabilität der Einnahmen zu optimieren.[26]

3.3.3. Sponsoring & Partnerschaften

Dieser Abschnitt widmet sich Sponsoring und Partnerschaften als alternatives Monetarisierungsmodell für mobile Anwendungen. Dabei wird das Grundprinzip des App-Sponsorings sowie dessen Bedeutung als integriertes Kommunikationsinstrument im digitalen Umfeld erläutert.

Grundprinzip von App-Sponsoring
App-Sponsoring ist als spezieller Anwendungsfall des Sponsorings innerhalb der Marketing- und Unternehmenskommunikation zu verstehen. Im Gegensatz zu klassischen Werbeformen oder rein altruistischen Fördermaßnahmen basiert Sponsoring grundsätzlich auf einem wechselseitigen Leistungsaustausch zwischen Sponsor und Gesponsertem. Ziel ist es, durch die Unterstützung eines Mediums, einer Organisation oder eines digitalen Produkts kommunikative und strategische Vorteile zu erzielen.[27]

Beim App-Sponsoring stellt die mobile Applikation die Plattform dar, auf der Unternehmen ihre Kommunikationsziele umsetzen können. Der Sponsor unterstützt dabei die Entwicklung, den Betrieb oder die inhaltliche Ausgestaltung einer App und erhält im Gegenzug definierte Kommunikations- und Präsentationsmöglichkeiten. Diese können beispielsweise in Form von Markenintegration, exklusiven Inhalten, Co-Branding oder einer prominenten Platzierung innerhalb der App erfolgen. Der bloße Erwerb von Werbeflächen steht dabei nicht im Vordergrund, vielmehr handelt es sich um ein alternatives Kommunikationsinstrument mit hohem Integrationsgrad.[27]

In Zeiten zunehmender medialer Reizüberflutung gewinnen solche Formen der Markenkommunikation an Bedeutung, da klassische Werbebotschaften häufig nicht mehr ausreichen, um eine nachhaltige Differenzierung gegenüber Wettbewerbern zu erreichen. App-Sponsoring ermöglicht es Unternehmen, Zielgruppen in einem alltäglichen, interaktiven Nutzungskontext zu erreichen und emotionale sowie funktionale Mehrwerte zu schaffen. Dadurch kann eine stärkere Bindung zwischen Marke und Nutzern aufgebaut werden.[27]

Ein zentrales Grundprinzip des App-Sponsorings ist die inhaltliche Passung zwischen Sponsor und App. Damit ein positiver Image-Transfer stattfinden kann, muss sich der Sponsor mit den Zielen, Werten und Inhalten der App identifizieren. Wird das Engagement lediglich als finanzielle Unterstützung ohne erkennbaren Bezug wahrgenommen, besteht die Gefahr, dass der kommunikative Effekt ausbleibt oder sogar negativ ausfällt. Die Auswahl geeigneter Sponsoring-Partnerschaften erfordert daher einen systematischen Planungs- und Entscheidungsprozess.[27]

3.3.4. Kostenpflichtige App-Modelle

Nun werden kostenpflichtige App-Modelle als Monetarisierungsstrategie betrachtet. Dabei werden sowohl der einmalige Kauf als auch Abonnement-Modelle und die damit verbundene Zahlungsbereitschaft der Nutzer erläutert.

Einmaliger Kauf
Beim diesem Modell bezahlen Nutzer vor dem Download der App einen fixen Betrag und erhalten anschließend die Erlaubnis die Applikation zu nutzen. Dieses Monetarisierungsmodell ist wohl eines der ältesten Ansätze im App-Markt und wird heute vor allem bei spezialisierten Anwendungen mit klarem Nutzen eingesetzt. Vorteile dieses Modells sind die transparente Preisstruktur sowie eine werbefreie Nutzung. Allerdings stellt der einmalige Kauf eine hohe Einstiegshürde dar, da viele Nutzer kostenlose Alternativen bevorzugen. Zudem ist das Umsatzpotenzial begrenzt, da pro Nutzer nur einmal Einnahmen generiert werden.[28]

Abonnement-Modelle
Abonnement-Modelle basieren auf wiederkehrenden Zahlungen, meist in monatlichen oder jährlichen Intervallen. Nutzer erhalten im Gegenzug fortlaufenden Zugriff auf Premium-Funktionen, exklusive Inhalte oder regelmäßig aktualisierte Services. Dieses Modell ermöglicht planbare und stabile Einnahmen für Entwickler und führt bei hoher Nutzerbindung zu einem hohen Customer Lifetime Value. Gleichzeitig erfordert es jedoch kontinuierliche Weiterentwicklung und regelmäßige Inhalte, da Nutzer ihr Abonnement jederzeit kündigen können, wenn der wahrgenommene Mehrwert sinkt.[28]

Zahlungsbereitschaft der Nutzer
Die Zahlungsbereitschaft der Nutzer hängt stark vom wahrgenommenen Nutzen und der Qualität der App ab. Faktoren wie Benutzerfreundlichkeit, Exklusivität der Funktionen, Aktualität der Inhalte sowie ein professioneller Gesamteindruck beeinflussen die Entscheidung, ob Nutzer bereit sind, für eine App zu bezahlen. Besonders erfolgreich sind kostenpflichtige Modelle, wenn Nutzer den Mehrwert bereits vor dem Kauf durch Testphasen oder eingeschränkte kostenlose Versionen kennenlernen können.[28]

3.3.5. In-App-Käufe

Als nächstes werden In-App-Käufe als flexibles Monetarisierungsmodell mobiler Anwendungen vorgestellt. Dabei werden die grundlegenden Konzepte von In-App-Purchases sowie deren Einsatzmöglichkeiten, insbesondere im Bereich der Personalisierung, erläutert.

Grundlagen von In-App-Purchases
In-App-Käufe ermöglichen es Nutzern, zusätzliche digitale Inhalte oder Funktionen direkt innerhalb der App zu erwerben. Dieses Modell wird häufig in Spielen, Produktivitäts-Apps und Content-Plattformen eingesetzt. Dabei unterscheidet man zwischen verbrauchbaren Käufen, wie virtueller Währung, und nicht-verbrauchbaren Käufen, wie dem dauerhaften Freischalten von Funktionen oder dem Entfernen von Werbung. Die Zahlungsabwicklung erfolgt über die integrierten Systeme der jeweiligen App-Stores, was Sicherheit und Standardisierung gewährleistet.[28]

Personalisierung
Ein häufiger Anwendungsbereich von In-App-Käufen ist die persönliche Gestaltung der App. Nutzer können gegen Bezahlung individuelle Anpassungen vornehmen, die keinen direkten funktionalen Vorteil bieten, jedoch das Nutzererlebnis verbessern. Beispiele für personalisierte Inhalte sind:

- Profilbilder
- Banner
- Schriftfarben und Schriftarten

Solche personalisierten Inhalte fördern die Bindung der Nutzer an die App und stellen insbesondere bei engagierten Nutzern eine effektive Einnahmequelle dar.[28]

3.3.6. Spenden & freiwillige Unterstützung

Der folgende Abschnitt behandelt Spenden und freiwillige Unterstützung als alternative Monetarisierungsform mobiler Anwendungen. Dabei wird aufgezeigt, wie Crowdfunding und In-App-Spenden zur Finanzierung, Ideenvalidierung und zum Aufbau einer engagierten Nutzerbasis beitragen können.

Spenden & Crowdfunding
Ein weiteres Einnahmemodell besteht darin, Nutzer um freiwillige finanzielle Unterstützung zu bitten, ohne ihnen Pflichten oder Käufe aufzuzwingen. Im Unterschied zu den Anderen

Einnahmemodellen (z. B. Werbung, In-App-Käufe oder Abonnements) basiert dieses Modell auf freiwilligen Beiträgen der Nutzer, welche die App oder ihre Entwickler aus Überzeugung unterstützen möchten.

Crowdfunding bietet die Möglichkeit, finanzielle Mittel direkt von Menschen zu erhalten, die an die Vision oder den Nutzen einer App glauben. Über Plattformen wie Kickstarter oder GoFundMe kann bereits vor oder während der Entwicklungsphase Kapital für die Umsetzung eines App-Projekts gesammelt werden. Dieses Finanzierungsmodell eignet sich besonders für Apps mit sozialen Anliegen oder kreativen Konzepten. Darüber hinaus trägt Crowdfunding nicht nur zur Finanzierung bei, sondern kann auch als Validierungsverfahren der App-Idee und zum Aufbau einer frühen, engagierten Nutzerbasis dienen.[29]

3.4. Technische Grundlagen der Anwendung

Dieser Abschnitt vermittelt die technischen Grundlagen der entwickelten Anwendung. Dabei werden sowohl zentrale Konzepte der mobilen App-Entwicklung als auch wesentliche Backend-, Infrastruktur- und Sicherheitsaspekte vorgestellt, die für den Aufbau moderner, skalierbarer und sicherer Anwendungen erforderlich sind.

3.4.1. Mobile App Entwicklung

Die Entwicklung mobiler Anwendungen spielt eine zentrale Rolle in der heutigen digitalen Welt. Unterschiedliche Plattformen wie iOS und Android stellen spezifische Anforderungen an Technologie, Performance und Benutzerfreundlichkeit. In diesem Abschnitt werden zentrale Frameworks, Konzepte und Techniken vorgestellt, die eine effiziente, plattformübergreifende Entwicklung ermöglichen.

3.4.1.0.1. React Native Framework

React Native ist ein plattformübergreifendes Framework zur Entwicklung mobiler Anwendungen, das von Meta Platforms (ehemals Facebook) und der Open-Source-Community entwickelt wird. Es ermöglicht, Anwendungen für verschiedene mobile Betriebssysteme wie iOS und Android zu erstellen, indem es die Programmiersprache JavaScript in Kombination mit nativen UI-Elementen nutzt [30].

Im Gegensatz zu klassischen nativen Entwicklungsansätzen, bei denen separate Codebasen für unterschiedliche Betriebssysteme notwendig sind, verfolgt React Native den Ansatz einer gemeinsamen Codebasis. Dabei werden Benutzeroberflächen nicht als Webview-Elemente gerendert, sondern über native Komponenten dargestellt, sodass die resultierenden Anwendungen sich in Look and Feel deutlich näher an nativen Apps orientieren [30].

React Native basiert auf dem populären JavaScript-Framework React, das ursprünglich für die Entwicklung von Benutzeroberflächen im Web entwickelt wurde. Die Integration dieser Technologie ermöglicht es Entwicklern, UI-Komponenten modular zu strukturieren und

wiederverwendbar zu machen, was die Wartung und Weiterentwicklung von Anwendungen erleichtert [30].

Ein technisches Charakteristikum von React Native ist der Einsatz der JavaScriptCore-Laufzeit sowie von Babel-Transpilern, die moderne JavaScript-Funktionen (z. B. ES6-Features wie Arrow-Funktionen oder async/await) unterstützen und zugleich die Kompatibilität mit unterschiedlichen Zielplattformen sicherstellen. Dadurch können Entwickler aktuelle Sprachstandards verwenden, ohne auf traditionelle plattformspezifische Sprachen wie Swift (für iOS) oder Kotlin/Java (für Android) angewiesen zu sein [30].

3.4.2. Backend-Technologien

Das Backend bildet das Rückgrat moderner Anwendungen und ist für Datenverarbeitung, Geschäftslogik, Sicherheit und Kommunikation mit Datenbanken verantwortlich. Unterschiedliche Technologien, Architekturen und Dienste ermöglichen es, skalierbare, performante und wartbare Systeme aufzubauen. In diesem Abschnitt werden zentrale Konzepte, Infrastrukturmodelle und Cloud-Ansätze vorgestellt, die in modernen Backend-Umgebungen zum Einsatz kommen.

3.4.2.0.1. APIs

Application Programming Interfaces sind Programmierschnittstellen, die es unterschiedlichen Softwaresystemen ermöglichen, miteinander zu kommunizieren. Sie definieren, wie Anfragen gestellt und wie Daten oder Funktionen zwischen Anwendungen ausgetauscht werden können, ohne dass die internen Abläufe der beteiligten Systeme bekannt sein müssen [31].

Eine API fungiert dabei als Vermittlungsschicht zwischen verschiedenen Softwarekomponenten. Sie legt fest, welche Funktionen oder Daten zur Verfügung stehen und in welcher Form diese genutzt werden dürfen. Durch diese klar definierten Schnittstellen wird eine strukturierte und kontrollierte Interaktion zwischen Frontend- und Backend-Systemen ermöglicht [31].

In der Softwareentwicklung bilden APIs eine wesentliche Grundlage für den Aufbau modularer Systeme. Sie unterstützen die Trennung von Zuständigkeiten zwischen einzelnen Systemkomponenten und tragen zur Wartbarkeit sowie Erweiterbarkeit von Anwendungen bei [31].

3.4.2.0.2. Datenbanken

Datenbanken sind digitale Speichersysteme zur organisierten Verwaltung von Informationen. Sie dienen dazu, große Mengen strukturierter und unstrukturierter Daten so zu speichern, dass Anwendungen, Benutzer und Automatisierungsprozesse effizient darauf zugreifen, diese verwalten und aktualisieren können. Eine Datenbank besteht dabei aus einem Repository zur Speicherung der Daten sowie aus Software-Komponenten, die den Zugriff, die Strukturierung und die Sicherheit dieser Daten steuern [32].

Die grundlegende Funktion einer Datenbank besteht darin, Daten nicht nur passiv zu speichern, sondern sie in einer Form bereitzustellen, die schnelle Abfragen, konsistente Verwaltung und kontrollierten Zugriff ermöglicht. Datenbanken bilden damit eine essentielle Grundlage moderner Anwendungen, da sie die zentrale Grundlage für die Verwaltung und Bereitstellung von Daten in Informationssystemen darstellen [32].

Dabei umfasst der Begriff „Datenbank“ nicht nur die gespeicherten Daten selbst, sondern auch die zugehörige Infrastruktur, die physische Speicherung und die Software-Komponenten einschließt, die Datenbankoperationen steuern und ausführen. Durch diese Struktur ermöglichen Datenbanken eine systematische Organisation großer Datenbestände, wodurch diese für Anwendungen adressierbar und nutzbar werden [32].

Datenbanken werden in vielfältigen Kontexten verwendet und sind integraler Bestandteil zahlreicher Softwarelösungen, da sie die grundlegende Datenverwaltung für Anwendungen unterstützen. Ohne Datenbanken wäre die zentrale Verwaltung großer Informationsmengen sowie deren strukturierter Zugriff, wie er für Web-Anwendungen, mobile Anwendungen und Backend-Systeme heute notwendig ist, nicht realisierbar [32].

3.4.2.0.3. Relationale Datenbanken

Relationale Datenbanken sind ein Datenbanktyp, bei dem Daten in Form von Tabellen mit Zeilen und Spalten strukturiert gespeichert werden. Dieses Modell basiert auf dem relationalen Datenbankmodell, bei dem jede Zeile einer Tabelle einen einzelnen Datensatz mit einer eindeutigen Kennung, dem sogenannten Primärschlüssel, darstellt, und jede Spalte ein Attribut des Datensatzes beschreibt. Durch gemeinsame Spalten zwischen verschiedenen Tabellen können Relationen zwischen Datensätzen hergestellt werden, was eine konsistente und strukturierte Verbindung von Daten über mehrere Tabellen hinweg ermöglicht.[33]

Im relationalen Modell sind die logischen Datenstrukturen wie Tabellen, Ansichten und Indizes von der physischen Speicherung getrennt, wodurch die Organisation und Verwaltung der Daten unabhängig von der physischen Lage erleichtert wird. Integritätsregeln sorgen dafür, dass die Daten konsistent und fehlerfrei bleiben, etwa indem doppelte Werte in Schlüsselfeldern verhindert werden.

Relationale Datenbanken sind weit verbreitet und werden in vielen Bereichen eingesetzt, beispielsweise zur Verwaltung von Bestellungen, Kundendaten oder anderen geschäftskritischen Informationen. Sie erlauben strukturierte Abfragen und Manipulationen der Daten mithilfe standardisierter Sprachen wie SQL (Structured Query Language), die auf dem relationalen Modell aufbauen.

3.4.2.0.4. Datenmodellierung und Normalisierung

Die Datenmodellierung in relationalen Datenbanksystemen umfasst neben der strukturellen Abbildung von Informationsbedarfen auch die Optimierung des Datenmodells zur Vermeidung unnötiger Redundanzen. Ein zentrales Konzept in diesem Zusammenhang ist die Normalisierung, bei der das Datenmodell so gestaltet wird, dass redundante Daten möglichst vermieden und damit verbundene semantische Probleme reduziert werden. Dabei wird ein

ursprünglich grob entworfenes Relationenschema durch eine Folge von Normalisierungsregeln so umgestaltet, dass die Daten konsistent und effizient gespeichert werden.[34]

Das Ziel der Normalisierung besteht primär darin, Wiederholungen gleicher Informationen innerhalb einer Datenbankstruktur zu minimieren und dadurch sogenannte Anomalien zu vermeiden. Anomalien treten insbesondere beim Einfügen, Ändern oder Löschen von Daten auf und können durch redundante Speicherung auftreten. Durch die Aufteilung eines Datenmodells in mehrere, logisch getrennte Tabellenstrukturen werden Abhängigkeiten besser kontrolliert und Redundanzen reduziert, was die Konsistenz der Daten erhöht.

Im Normalisierungsprozess werden Datenstrukturen (Tabellen) schrittweise so angepasst, dass sie bestimmten Normalformen entsprechen. Üblicherweise wird dieser Prozess bis zur dritten Normalform durchgeführt, da dadurch die meisten redundanzbedingten Probleme eliminiert werden, während die Komplexität des Datenmodells in einem praktikablen Rahmen bleibt. Die Anwendung von Normalisierungsregeln stellt somit einen wichtigen Schritt bei der Modellierung relationaler Datenbanken dar und bildet die Grundlage für eine konsistente und wartbare Datenstruktur.

3.4.2.0.5. Echtzeit-Daten

Echtzeit-Daten (engl. „Real Time Data“) sind Informationen, die unmittelbar nach ihrer Erfassung gesammelt, verarbeitet und zur Verfügung gestellt werden. Im Gegensatz zu klassischen periodisch aktualisierten Daten, die in festen Intervallen erfasst und verarbeitet werden, zeichnen sich Echtzeit-Daten dadurch aus, dass sie nahezu ohne Verzögerung bereitgestellt werden und sofortige Reaktionen auf Ereignisse oder Veränderungen ermöglichen. Dadurch wird es möglich, Entscheidungen auf Basis aktueller Informationen zu treffen und dynamisch auf Veränderungssituationen zu reagieren.[35]

Die Entwicklung von Echtzeit-Daten ist eng mit der technischen Weiterentwicklung leistungsfähiger Systeme und schneller Netzwerke verbunden, die es erlauben, große Datenmengen direkt zu erfassen und nahezu in Echtzeit zu verarbeiten. Anwendungen von Echtzeit-Daten finden sich in diversen Bereichen wie beispielsweise der Finanzbranche, wo aktuelle Marktinformationen für Handelsentscheidungen essentiell sind, oder in der Logistik, wo kontinuierliche Standortdaten zur Optimierung von Lieferprozessen genutzt werden.

Technologien zur Unterstützung von Echtzeit-Daten bestehen aus Systemen, die Datenströme kontinuierlich analysieren und verarbeiten. Dazu zählen unter anderem Cloud-Computing-Plattformen und spezialisierte Streaming-Technologien, welche Latenzzeiten minimieren und die Effizienz der Datenverarbeitung steigern. Durch diese technischen Ansätze wird die schnelle Verarbeitung großer Datenströme ermöglicht, wodurch Echtzeit-Daten in vielfältigen praktischen Kontexten nutzbar werden.

3.4.2.0.6. Vertikale vs. horizontale Skalierung

Skalierung ist ein zentrales Konzept in IT-Architekturen und beschreibt, wie Systeme mit steigender Last umgehen können. Dabei wird unterschieden zwischen der **vertikalen Skalierung**, bei der die Leistungsfähigkeit einzelner Maschinen erhöht wird, und der **horizontalen**

Skalierung, bei der zusätzliche Maschinen oder Knoten zum System hinzugefügt werden [36].

Vertikale Skalierung

Vertikale Skalierung, auch als „Scaling Up“ (Skalierung nach oben) bezeichnet, beschreibt den Prozess der Erhöhung der Leistungsfähigkeit einer einzelnen Maschine. Dabei werden die Ressourcen eines bestehenden Systems erweitert, indem beispielsweise die Anzahl der CPUs, der Arbeitsspeicher oder der Speicherplatz vergrößert wird. Dadurch kann die Maschine höhere Arbeitslasten und mehr Anfragen verarbeiten, ohne dass zusätzliche Knoten in ein System eingeführt werden müssen. Vertikale Skalierung wird häufig genutzt, um innerhalb der Grenzen eines einzelnen Servers die Leistungsfähigkeit zu steigern und ist insbesondere dann sinnvoll, wenn ein einzelner Knoten den größten Teil der Arbeitslast übernimmt oder die Architektur einer Anwendung nicht für verteilte Systeme ausgelegt ist.[36]

Horizontale Skalierung

Horizontale Skalierung, auch als „Scaling Out“ (Skalierung nach außen) bezeichnet, bezeichnet den Ansatz, zusätzliche Maschinen oder Knoten zu einem System hinzuzufügen, um die Arbeitslast über mehrere Einheiten zu verteilen. Bei diesem Modell werden weitere virtuelle Maschinen oder physische Server in einen Cluster integriert, um die Gesamtkapazität zu erhöhen. Die Lastverteilung erfolgt dabei über spezialisierte Ressourcenverwaltungs-Software, die sicherstellt, dass Anfragen effizient auf die vorhandenen Knoten verteilt werden. Horizontale Skalierung ermöglicht es, die Systemkapazität bei Bedarf dynamisch zu erweitern, indem zusätzliche Einheiten eingefügt werden, was insbesondere in verteilten Systemen und cloud-basierten Architekturen Anwendung findet.[36]

3.4.2.0.7. Autoscaling und Lastverteilung

Autoscaling bezeichnet einen Mechanismus in Cloud- und IT-Infrastrukturen, der es ermöglicht, Rechenressourcen automatisch an die aktuelle Systemlast anzupassen. Dabei werden bei steigender Arbeitslast zusätzliche Ressourcen bereitgestellt und bei sinkender Last wieder freigegeben, ohne dass ein manuelles Eingreifen durch Administratoren erforderlich ist. Ziel des Autoscalings ist es, Leistungsfähigkeit und Effizienz der Infrastruktur dynamisch zu optimieren, indem Überlastungen vermieden und Ressourcenverschwendung reduziert werden.[37]

Lastverteilung (engl. Load Balancing) beschreibt den Prozess, eingehende Anfragen gleichmäßig auf mehrere Server oder Ressourcen zu verteilen. Durch die Verteilung der Arbeitslast auf verschiedene Knoten kann die Leistungsfähigkeit und Verfügbarkeit eines Systems gesteigert werden. Lastverteilende Komponenten analysieren den aktuellen Zustand der Server und leiten Anfragen so weiter, dass keine einzelne Ressource übermäßig belastet wird.[37]

In Kombination tragen Autoscaling und Lastverteilung dazu bei, Systeme flexibel und robust gegenüber Lastschwankungen zu machen. Während das Autoscaling die Anzahl oder Leistungsfähigkeit der Ressourcen anpasst, sorgt die Lastverteilung dafür, dass die vorhandenen Ressourcen effizient genutzt werden, indem sie die Anfragen gleichmäßig verteilt. Zusammen bilden diese Konzepte zentrale Bausteine moderner skalierbarer und hochverfügbarer IT-Architekturen.[37]

3.4.2.0.8. Latenzoptimierung und Durchsatzsteigerung

Latenz bezeichnet in der Informatik die Verzögerungszeit, die ein Datenpaket benötigt, um von einem Punkt zum anderen übertragen zu werden. Sie wird üblicherweise in Millisekunden gemessen und beeinflusst maßgeblich die Reaktionsfähigkeit von Netzwerken und Systemen, da sie die Zeitspanne zwischen Anforderung und Antwort beschreibt. Eine niedrige Latenz ist insbesondere bei interaktiven oder zeitkritischen Anwendungen wie Videokonferenzen, Online-Spielen oder Echtzeitübertragungen entscheidend.[38]

Durchsatz definiert die Menge an Daten, die innerhalb eines bestimmten Zeitraums erfolgreich übertragen werden kann. Er wird meist in Bits pro Sekunde (bps) gemessen und gibt an, wie viel Datenvolumen ein Netzwerk oder System in einer definierten Zeitspanne verarbeiten kann. Ein hoher Durchsatz ist essenziell, um große Datenmengen effizient zu übertragen und die Leistungsfähigkeit von Netzwerken zu bewerten.[38]

Techniken zur Latenzoptimierung konzentrieren sich darauf, die Verzögerungszeiten bei der Datenübertragung zu reduzieren. Dazu gehören unter anderem der Einsatz schnellerer Übertragungswege, optimierte Routing-Algorithmen oder die Verringerung von Verarbeitungszeiten in Netzwerkknoten. Solche Maßnahmen tragen dazu bei, die benötigte Zeit bis zur Zustellung von Datenpaketen zu verkürzen und somit die Geschwindigkeit und Reaktionsfähigkeit eines Systems zu erhöhen.[38]

Die Durchsatzsteigerung wird erreicht, indem die Effizienz der Datenübertragung erhöht wird. Faktoren wie die Netzwerkarchitektur, die verfügbare Bandbreite, die Paketgröße sowie die Leistungsfähigkeit der beteiligten Hardware beeinflussen den Durchsatz. Durch den Einsatz leistungsfähiger Hardware, effizienter Protokolle und geeigneter Übertragungsstrategien kann die Datenrate gesteigert werden, wodurch mehr Daten in kürzerer Zeit verarbeitet werden können.[38]

3.4.2.0.9. Backend-as-a-Service (BaaS)

Backend-as-a-Service (BaaS) bezeichnet ein cloudbasiertes Backend-Plattformmodell, das Entwicklungswerkzeuge und Dienste bereitstellt, um Backend-Funktionalitäten für Anwendungen schnell und ohne eigenen Serveraufwand bereitzustellen. Im Mittelpunkt dieses Modells steht die Bereitstellung einer Infrastruktur, die typische Backend-Aufgaben übernimmt, wie etwa Datenverwaltung, Authentifizierung, API-Generierung und weitere Dienste, ohne dass Entwickler diese Komponenten selbst implementieren müssen [39].

Ein Beispiel für eine solche Plattform ist Supabase, eine Open-Source-BaaS-Lösung, die auf einer PostgreSQL-Datenbank basiert und Werkzeuge zur Backend-Entwicklung zusammenführt. Supabase stellt Entwicklern eine Reihe von integrierten Tools zur Verfügung, darunter [39]:

- **PostgreSQL-Datenbank:** Als zentrales Speichersystem bildet die relationale PostgreSQL-Datenbank den Kern von Supabase und dient zur strukturierten Verwaltung von Anwendungsdaten.
- **Studio (Dashboard):** Ein offenes Dashboard, das die Verwaltung der Datenbankservices und Projekte ermöglicht.

- **Authentifizierungsdienst (GoTrue):** Eine API-basierte Komponente zur Benutzerverwaltung und zur Ausstellung von Zugangstoken.
- **Automatisch generierte APIs (PostgREST):** Supabase erzeugt aus der Datenbank heraus RESTful-APIs, die die Interaktion mit Daten über standardisierte Schnittstellen erlauben.
- **Realtime-Funktionalität:** Diensten zur Verwaltung von Echtzeit-Datenübertragungen und -Präsenz mittels skalierbarer WebSocket-Technologien.
- **Speicher-API:** Ein Service zur Verwaltung großer Dateien und Objekte.
- **Edge Functions (Deno):** Eine moderne Laufzeitumgebung für serverlose Funktionen in JavaScript/TypeScript.
- **Datenbankmanagement-Tools:** RESTful-APIs zum Verwalten der Datenbankstruktur, Tabellen, Rollen und Abfragen.
- **Supavisor und API-Gateway-Komponenten:** Unterstützung für Pooling und API-Steuerung innerhalb der Cloud-Architektur.

Supabase ermöglicht es Entwicklern, Backend-Funktionalität „out of the box“ zu nutzen und so Anwendungen schnell zu entwickeln und bereitzustellen. Die Plattform unterstützt dabei verschiedene Frameworks für Web- und mobile Anwendungen, wodurch eine breite Integration mit Frontend-Technologien möglich ist [39].

Insgesamt bietet BaaS-Plattformen wie Supabase eine abstrahierte Backend-Schicht, die vielen klassischen Backend-Aufgaben übernimmt und Entwickler von der Notwendigkeit befreit, eigene Backend-Infrastruktur manuell aufzusetzen oder zu warten [39].

3.4.2.0.10. Auth-Systeme

Authentifizierungssysteme sind Verfahren zur Überprüfung der Identität von Benutzern oder Systemen, bevor diesen Zugriff auf geschützte Ressourcen gewährt wird. Moderne Authentifizierungsmechanismen gehen über die klassische Passwortabfrage hinaus und beinhalten unterschiedliche Ansätze, um Sicherheit und Benutzerfreundlichkeit gleichzeitig zu erhöhen. Dabei wird oftmals eine Kombination verschiedener Techniken eingesetzt, um das Risiko unbefugter Zugriffe zu minimieren.[40]

Multi-Faktor-Authentifizierung (MFA): Ein zentraler Ansatz moderner Authentifizierung ist die Multi-Faktor-Authentifizierung, bei der mindestens zwei unabhängige Faktoren zur Identitätsprüfung herangezogen werden. Diese Faktoren lassen sich üblicherweise in drei Kategorien einteilen:

- Wissen (z. B. Passwort oder PIN)
- Besitz (z. B. Smartphone oder Hardware-Token)
- Sein (biometrische Merkmale wie Fingerabdruck oder Gesichtserkennung)

Durch die Kombination dieser Faktoren wird die Sicherheit gegenüber einem einzelnen Faktor wie einem Passwort deutlich erhöht, da ein Angreifer mehrere unabhängige Sicherheitsmerkmale überwinden müsste.[40]

Biometrische Authentifizierung: Die biometrische Authentifizierung nutzt einzigartige körperliche Merkmale zur Identifikation eines Benutzers. Zu den gängigen Verfahren zählen Fingerabdruckscanner, Gesichtserkennung, Iris- oder Retina-Scans sowie Stimmerkennung. Diese Methoden gelten aufgrund ihrer individuellen Eigenschaften als schwer zu fälschen und bieten einen zusätzlichen Sicherheitsgrad, insbesondere in mobilen oder gerätebasierten Systemen.[40]

Einmalpasswort (OTP): Weitere Methoden umfassen die Authentifizierung mittels Einmalpasswort, bei der zeitlich begrenzte Codes verwendet werden, die z. B. durch Software-Token, SMS oder spezielle Hardware-Token generiert werden. Diese Verfahren kommen häufig als zusätzliche Sicherheitsstufe in Kombination mit anderen Faktoren zum Einsatz.[40]

Zertifikatsbasierte Authentifizierung: Digitale Zertifikate, die von vertrauenswürdigen Zertifizierungsstellen ausgestellt werden, bestätigen die Identität eines Benutzers oder Geräts. Dies findet insbesondere im Unternehmenskontext Anwendung, beispielsweise zur Absicherung von Netzwerkzugängen oder VPN-Verbindungen.[40]

FIDO2 und WebAuthn: Moderne Standards wie FIDO2 und WebAuthn ermöglichen passwortlose Authentifizierungsprozesse auf Basis von Public-Key-Kryptographie. Benutzer können sich damit sicher anmelden, ohne traditionelle Passwörter zu verwenden, indem kryptografische Schlüsselpaare genutzt werden.[40]

Social Login: Die Authentifizierung über soziale Netzwerke wie Google, Facebook oder LinkedIn erlaubt Benutzern, bestehende Identitäten zur Anmeldung zu nutzen. Dies vereinfacht den Anmeldeprozess, da keine neuen Zugangsdaten erstellt werden müssen.[40]

Kontext- und risikobasierte Authentifizierung: Moderne Authentifizierungssysteme können zudem Informationen über Standort, Gerätetyp oder Benutzerverhalten berücksichtigen, um Zugriffsrisiken zu bewerten. Dadurch lassen sich adaptive Sicherheitsmaßnahmen implementieren, die je nach Kontext unterschiedliche Authentifizierungsanforderungen stellen.[40]

3.4.2.0.11. Verschlüsselung

Verschlüsselung bezeichnet den Vorgang, bei dem Daten mithilfe eines Algorithmus in eine codierte Form umgewandelt werden, sodass sie für Unbefugte nicht mehr lesbar sind. Dieser Prozess hat zum Ziel, die Vertraulichkeit und Integrität von Informationen zu gewährleisten, indem nur autorisierte Parteien mit dem passenden Schlüssel die verschlüsselten Daten wieder in ihre ursprüngliche Form zurückverwandeln können. Verschlüsselung ist ein grundlegender Bestandteil moderner Datensicherheit und wird sowohl beim Speichern als auch bei der Übertragung sensibler Informationen angewendet.[41]

Bei der Verschlüsselung werden lesbare Daten (Klartext) in einen unlesbaren Chiffretext überführt. Diese Umwandlung erfolgt mithilfe eines kryptografischen Schlüssels, der in Verbindung mit dem gewählten Algorithmus bestimmt, wie die Umwandlung stattfindet. Je komplexer der Schlüssel und der Algorithmus gestaltet sind, desto schwieriger ist es für unbefugte Dritte, den Chiffretext zu entschlüsseln.[41]

Es existieren unterschiedliche Verschlüsselungsverfahren, die je nach Anwendungsfall eingesetzt werden können. Zu den grundlegenden Unterscheidungen gehören symmetrische Verfahren,

bei denen derselbe Schlüssel sowohl für die Verschlüsselung als auch für die Entschlüsselung verwendet wird, und asymmetrische Verfahren, bei denen ein Schlüssel zur Verschlüsselung und ein anderer Schlüssel zur Entschlüsselung zum Einsatz kommt. Beide Verfahren spielen eine zentrale Rolle bei der Absicherung digitaler Kommunikation.[41]

Verschlüsselung wird in vielen Bereichen eingesetzt, um Daten vor unbefugtem Zugriff zu schützen, etwa bei der Sicherung von Nachrichtenübertragungen, dem Schutz gespeicherter personenbezogener Daten oder der Absicherung von Online-Transaktionen. Durch die Anwendung geeigneter Verschlüsselungstechniken wird sichergestellt, dass selbst bei einem Abfangen der Daten durch Dritte die Informationen nicht ohne Wissen über den Schlüssel verständlich sind.[41]

3.4.2.0.12. Datenschutz

Datenschutz bezeichnet den Schutz personenbezogener Daten vor unbefugtem Zugriff, Missbrauch oder Verlust. Ziel ist es, die Privatsphäre von Personen zu wahren und sicherzustellen, dass Informationen nur in zulässiger Weise verarbeitet werden. Im digitalen Kontext betrifft Datenschutz insbesondere die Erhebung, Speicherung, Verarbeitung und Übertragung von Daten durch Anwendungen, Dienste oder Organisationen.[42]

Für mobile Apps bedeutet Datenschutz, dass Nutzer über Art, Umfang und Zweck der Datenverarbeitung informiert werden müssen. Dazu gehören unter anderem Hinweise darauf, welche Daten gesammelt werden, wie lange sie gespeichert werden und wer Zugriff darauf hat. Transparenz und rechtliche Vorgaben, wie sie in der Datenschutz-Grundverordnung (DSGVO) festgelegt sind, bilden die Grundlage für vertrauenswürdige Anwendungen.[42]

Mobile Anwendungen müssen geeignete technische und organisatorische Maßnahmen implementieren, um die Sicherheit der Daten zu gewährleisten. Dazu zählen Verschlüsselung, Zugriffsbeschränkungen, Anonymisierung oder Pseudonymisierung, um sicherzustellen, dass personenbezogene Informationen vor Missbrauch geschützt sind. Datenschutz umfasst somit sowohl die rechtlichen Rahmenbedingungen als auch die praktischen Maßnahmen zur Sicherung sensibler Daten.[42]

3.5. Hosting- und Infrastrukturmodelle

Hosting- und Infrastrukturmodelle im Kontext moderner IT-Systeme beschreiben, wo und wie Rechenressourcen, Speicher und Anwendungen betrieben werden. Sie bilden die Grundlage für Verfügbarkeit, Skalierbarkeit und Sicherheit moderner Anwendungen. Grundsätzlich lassen sich drei zentrale Architekturansätze unterscheiden: On-Premise-Infrastrukturen, Cloud-Architekturen und Hybrid-Architekturen [43, 44].

3.5.1. On-Premise, Cloud und Hybrid-Architekturen

Bei einer On-Premise-Architektur werden sämtliche IT-Ressourcen innerhalb eines Unternehmens betrieben. Die notwendige Hardware wie Server, Storage-Systeme und Netzwerkkom-

ponenten befindet sich in eigenen Räumlichkeiten oder Rechenzentren und wird vollständig selbst verwaltet. Dieses Modell ermöglicht ein hohes Maß an Kontrolle über Daten, Systeme und Sicherheitsmechanismen, ist jedoch mit hohem Aufwand für Anschaffung, Wartung und Betrieb verbunden [43].

Cloud-Architekturen basieren hingegen auf der Bereitstellung von IT-Ressourcen durch externe Anbieter über das Internet. Rechenleistung, Speicher und Dienste werden bedarfsgerecht zur Verfügung gestellt und zentral vom Provider verwaltet. Dadurch profitieren Organisationen von hoher Skalierbarkeit und Flexibilität, da Ressourcen dynamisch angepasst werden können und keine eigene Infrastruktur im selben Umfang betrieben werden muss [43].

Hybrid-Architekturen kombinieren On-Premise- und Cloud-Ansätze. Bestimmte Systeme oder Daten verbleiben lokal, während andere Komponenten in einer Cloud-Umgebung betrieben werden. Dadurch können Unternehmen sowohl die Kontrolle und Compliance-Vorteile lokaler Infrastruktur als auch die Skalierbarkeit und Effizienz von Cloud-Diensten nutzen. Hybrid-Cloud-Architekturen gewinnen besonders im Rahmen der digitalen Transformation an Bedeutung, da sie bestehende Systeme mit modernen Cloud-Technologien verbinden und eine flexible Verteilung von Anwendungen und Daten ermöglichen [44].

3.5.2. IaaS, PaaS, FaaS und Serverless

Cloud-Dienste lassen sich je nach Abstraktionsgrad in verschiedene Service-Modelle einteilen. Diese Modelle unterscheiden sich vor allem darin, wie viel Verantwortung der Cloud-Provider übernimmt und wie viel Verwaltung beim Betreiber verbleibt.

Infrastructure as a Service (IaaS) stellt grundlegende Infrastrukturressourcen wie virtuelle Maschinen, Speicher und Netzwerke bereit. Nutzer können diese Ressourcen flexibel konfigurieren und skalieren, ohne physische Hardware zu betreiben. Der Provider verwaltet dabei die zugrundeliegende Infrastruktur, während Konfiguration, Betriebssystem und Anwendungen in der Verantwortung der Nutzer liegen [45].

Platform as a Service (PaaS) bietet zusätzlich eine vollständig verwaltete Plattform zur Entwicklung und Bereitstellung von Anwendungen. Neben Infrastruktur stellt der Provider Laufzeitumgebungen, Middleware und häufig auch Entwicklungs- und Deployment-Tools bereit. Dadurch reduziert sich der administrative Aufwand deutlich, da sich Entwickler stärker auf die Anwendung selbst konzentrieren können [46].

Function as a Service (FaaS) ist ein ereignisgetriebenes Modell, bei dem einzelne Funktionen (Code-Snippets) automatisch ausgeführt werden, sobald ein bestimmtes Event eintritt (z. B. Request, Trigger, Message). Skalierung und Ressourcenmanagement erfolgen automatisch durch den Provider, wodurch sich FaaS besonders für klar abgegrenzte Logik und Microservice-Szenarien eignet [47].

Serverless Computing beschreibt ein umfassenderes Paradigma, bei dem Server weiterhin existieren, aber vollständig durch den Provider verwaltet werden. Entwickler betreiben keine Serverinstanzen aktiv, sondern nutzen verwaltete Services (z. B. Datenbanken, APIs, Event-Systeme) und stellen Funktionalität über FaaS oder ähnliche Mechanismen bereit. Ziel ist es, Betriebsaufwand zu minimieren und gleichzeitig automatisch skalierbare Architekturen zu ermöglichen [48].

3.5.3. Autoscaling und Lastverteilung

Autoscaling bezeichnet die automatische Anpassung von Ressourcen an die aktuelle Systemlast. Bei hoher Auslastung werden zusätzliche Instanzen bereitgestellt, bei sinkender Last werden Ressourcen reduziert. Dadurch können Leistung und Kosten effizienter gesteuert werden, da Über- oder Unterprovisionierung vermieden wird [37].

Lastverteilung (Load Balancing) beschreibt die Verteilung eingehender Anfragen auf mehrere Server oder Instanzen. Ein Load Balancer sorgt dafür, dass keine einzelne Ressource überlastet wird und dass Ausfälle einzelner Komponenten besser abgefangen werden können. In Kombination bilden Autoscaling und Load Balancing zentrale Bausteine moderner, hochverfügbarer und skalierbarer Systeme [37].

3.5.4. Latenzoptimierung und Durchsatzsteigerung

Für die Performance von Anwendungen sind insbesondere Latenz und Durchsatz entscheidend. Latenz beschreibt die Verzögerungszeit zwischen Anfrage und Antwort und ist vor allem bei interaktiven Anwendungen relevant. Durchsatz bezeichnet die Menge an Daten, die in einem bestimmten Zeitraum übertragen oder verarbeitet werden kann [38].

Latenzoptimierung zielt darauf ab, Verzögerungen zu reduzieren, beispielsweise durch optimierte Netzwerkrouten, Caching, kürzere Verarbeitungswege oder geografisch näher platzierte Ressourcen. Durchsatzsteigerung wird erreicht, indem Bandbreite, Parallelisierung, effiziente Protokolle oder leistungsfähigere Komponenten eingesetzt werden. Beide Aspekte beeinflussen direkt, wie schnell und stabil Systeme unter Last reagieren [38].

3.6. Skalierungsstrategie

Um eine bestehende App-Infrastruktur an stark wachsende Nutzerzahlen anzupassen, müssen konkrete technologische Entscheidungen getroffen werden. Dabei stellt sich nicht nur die Frage *wie* skaliert wird (vertikal oder horizontal), sondern insbesondere *mit welchen Diensten und Infrastrukturmodellen* die Skalierung technisch umgesetzt werden soll. Im Folgenden werden zentrale Cloud- und Backend-Ansätze gegenübergestellt und hinsichtlich ihrer Eignung für kurzfristiges sowie langfristiges Nutzerwachstum bewertet.

1. On-Premise vs. Cloud-Infrastruktur

Eine klassische On-Premise-Architektur bietet volle Kontrolle über Hardware und Daten, ist jedoch bei starkem Wachstum unflexibel. Neue Server müssen physisch beschafft, installiert und konfiguriert werden. Dies verursacht hohe Vorlaufzeiten und Investitionskosten.

Cloud-Infrastrukturen (z. B. AWS, Google Cloud, Microsoft Azure) ermöglichen dagegen die dynamische Bereitstellung von Ressourcen. Virtuelle Maschinen oder Container können innerhalb weniger Minuten gestartet werden. Für eine wachsende App ist die Cloud daher deutlich besser geeignet, da Skalierung nahezu in Echtzeit erfolgen kann.

Bewertung: Für eine App mit potenziell stark steigender Nutzerzahl ist eine Cloud-basierte Infrastruktur klar vorzuziehen, da sie flexible Skalierung ohne hohe Anfangsinvestitionen erlaubt.

2. IaaS vs. PaaS vs. BaaS

Bei der Auswahl des Cloud-Ansatzes existieren unterschiedliche Abstraktionsebenen:

Infrastructure as a Service (IaaS): Hier werden virtuelle Server bereitgestellt. Entwickler verwalten Betriebssystem, Runtime und Deployment selbst. Vorteil ist maximale Kontrolle. Nachteil ist hoher administrativer Aufwand. Skalierung erfolgt über zusätzliche Instanzen und Load Balancer.

Platform as a Service (PaaS): Der Anbieter übernimmt Betriebssystem und Laufzeitumgebung. Deployment erfolgt automatisiert (z. B. per Git-Push). Skalierung kann oft per Klick oder automatisch konfiguriert werden. Der Verwaltungsaufwand ist geringer als bei IaaS.

Backend as a Service (BaaS): Plattformen wie Supabase oder Firebase stellen Datenbank, Authentifizierung, Realtime-Funktionen und APIs direkt bereit. Skalierung wird größtenteils vom Anbieter übernommen. Entwickler konzentrieren sich auf die App-Logik.

Vergleich:

- IaaS: maximale Flexibilität, geeignet für komplexe Systeme mit hoher Individualisierung
- PaaS: gute Balance zwischen Kontrolle und Automatisierung
- BaaS: schnellste Skalierung mit minimalem Infrastrukturaufwand

Empfehlung: Für eine wachsende App in frühen bis mittleren Phasen ist BaaS oder PaaS besonders geeignet, da Skalierungsmechanismen bereits integriert sind. Bei sehr großen Systemen mit speziellen Anforderungen kann langfristig ein Wechsel zu IaaS sinnvoll sein.

3. Serverbasierte Architektur vs. Serverless

Klassische serverbasierte Architektur: Backend läuft auf dauerhaft aktiven Serverinstanzen. Skalierung erfolgt durch Hinzufügen weiterer Instanzen (Autoscaling). Vorteil ist konstante Performance und volle Kontrolle. Nachteil sind laufende Kosten auch bei geringer Nutzung.

Serverless / Function-as-a-Service (FaaS): Code wird ereignisgesteuert ausgeführt. Ressourcen werden automatisch skaliert. Es fallen nur Kosten bei tatsächlicher Nutzung an. Ideal bei stark schwankender Last.

Vergleich:

- Serverbasiert: stabil bei konstant hoher Last
- Serverless: flexibel bei stark variabler Last

Empfehlung: Für Apps mit unvorhersehbarem Wachstum oder saisonalen Peaks ist Serverless besonders geeignet. Bei dauerhaft hoher Nutzerzahl kann eine hybride Lösung sinnvoll sein.

4. Datenbankstrategien

Single-Instance-Datenbank: Einfach, aber bei hoher Last schnell überfordert.

Read-Replicas: Leseanfragen werden auf mehrere Replikate verteilt. Geeignet bei hohem Leseaufkommen.

Sharding: Aufteilung großer Datenmengen auf mehrere Datenbankinstanzen. Geeignet bei extrem großen Nutzerzahlen.

Relationale vs. NoSQL-Datenbanken: Relationale Systeme bieten hohe Konsistenz. NoSQL-Systeme sind oft horizontal besser skalierbar.

Empfehlung: Bei moderatem Wachstum reichen Read-Replicas aus. Bei massivem Wachstum ist Sharding oder ein verteilter Datenbankansatz notwendig.

5. Organisatorische Skalierungsstrategien

Technische Infrastruktur allein genügt nicht. Organisatorisch sind folgende Maßnahmen notwendig:

- Einführung von Monitoring-Systemen zur Lastüberwachung
- Regelmäßige Lasttests
- Automatisierte Deployments (CI/CD)
- Klare Trennung von Entwicklungs-, Test- und Produktionsumgebung
- Skalierbare Teamstrukturen (z. B. Aufteilung in Backend-, Infrastruktur- und Feature-Teams)

Gesamtbewertung für eine wachsende App

Für eine moderne mobile App mit potenziell stark steigender Nutzerzahl ist folgende Kombination besonders geeignet:

- Cloud-basierte Infrastruktur statt On-Premise
- PaaS oder BaaS in frühen Phasen
- Autoscaling + Load Balancing
- Caching und Datenbank-Replikation
- Kontinuierliches Monitoring und automatisierte Deployments

Zusammenfassend lässt sich feststellen, dass kurzfristige Skalierung vor allem durch Cloud-Ressourcen, Autoscaling und Replikation erreicht wird, während langfristige Stabilität durch architektonische Modularisierung, optimierte Datenbankstrategien und organisatorische Prozesse gewährleistet wird. Die optimale Lösung hängt dabei vom Entwicklungsstand, Budget und erwarteten Nutzerwachstum der jeweiligen App ab.

4. Dokumentation der Implementierung

In diesem Kapitel wird die technische Umsetzung des Spiels *WoSamma* dokumentiert. Dabei werden die verwendete Systemumgebung, eingesetzte Technologien sowie die grundlegenden Konzepte der Implementierung beschrieben, um einen strukturierten Überblick über die Realisierung der Anwendung zu geben.

4.1. Systemumgebung

Das Spiel WoSamma ist eine mobile Anwendung für iOS und Android und wurde mit React Native und TypeScript entwickelt. Die Entwicklung erfolgte unter Windows mithilfe von Visual Studio Code. Für das Veröffentlichen der Anwendung auf iOS-Geräten sowie für abschließende Tests und Fehlerbehebungen wurde Xcode unter macOS in Verbindung mit einem Apple-Developer-Konto verwendet. Die Tests für Android wurden mit Android Studio auf Emulatoren durchgeführt. Die Anwendung ist für eine zukünftige Veröffentlichung im Apple App Store sowie im Google Play Store vorgesehen.

4.1.1. Verwendete Technologien

- **React Native** – Framework zur Entwicklung plattformübergreifender mobiler Anwendungen für iOS und Android.
- **TypeScript** – Typsichere Erweiterung von JavaScript, die bewusst anstelle von JavaScript gewählt wurde, um die Wartbarkeit, Lesbarkeit und Codequalität der Anwendung zu erhöhen.
- **Supabase (PostgreSQL)** – Backend-as-a-Service, das zentrale Funktionen wie Authentifizierung, Echtzeitdaten, Datenbankverwaltung sowie eine REST-API bereitstellt und damit den Entwicklungsaufwand deutlich reduziert.
- **Google Maps API** – API zur Verarbeitung und Darstellung geografischer Daten sowie zur Positionsbestimmung innerhalb des Spiels.
- **Google Street View API** – API zur Integration von 360-Grad-Street-View-Bildern aus ganz Österreich, welche die Grundlage für das GeoGuessr-ähnliche Spielkonzept bildet.
- **GitHub** – Plattform zur Versionsverwaltung, die eine strukturierte Zusammenarbeit im Team sowie die kontinuierliche Aktualität des Quellcodes sicherstellt.
- **Jira** – Tool zur Projektplanung und Aufgabenverwaltung, das im Rahmen einer agilen Vorgehensweise für die Diplomarbeit eingesetzt wurde.
- **Figma** – Design- und Prototyping-Tool zur Erstellung von Mockups und interaktiven Prototypen, welche als visuelle Grundlage für die Benutzeroberfläche dienten.

4.2. Implementierung

Dieses Kapitel beschreibt die technische Umsetzung des Spiels *WoSamma*. Dabei wird auf den Aufbau der Anwendungsarchitektur, die Implementierung des Frontends, die Anbindung des Backends sowie auf die zentrale Spiellogik eingegangen. Der Fokus liegt auf den verwendeten Konzepten, der Struktur des Quellcodes und den getroffenen technischen Entscheidungen.

4.2.1. Architektur der Anwendung

Die Anwendung *WoSamma* folgt einer klaren Trennung zwischen Frontend, Backend und externen Diensten. Das Frontend wurde mit React Native umgesetzt und ist für die Benutzeroberfläche, die Spiellogik sowie die Interaktion mit dem Benutzer verantwortlich. Das Backend basiert auf Supabase und übernimmt Aufgaben wie Authentifizierung, Datenpersistenz und Echtzeitkommunikation. Externe APIs, wie die Google Maps und Google Street View API, werden zur Bereitstellung von geografischen Daten und 360-Grad-Bildern verwendet.

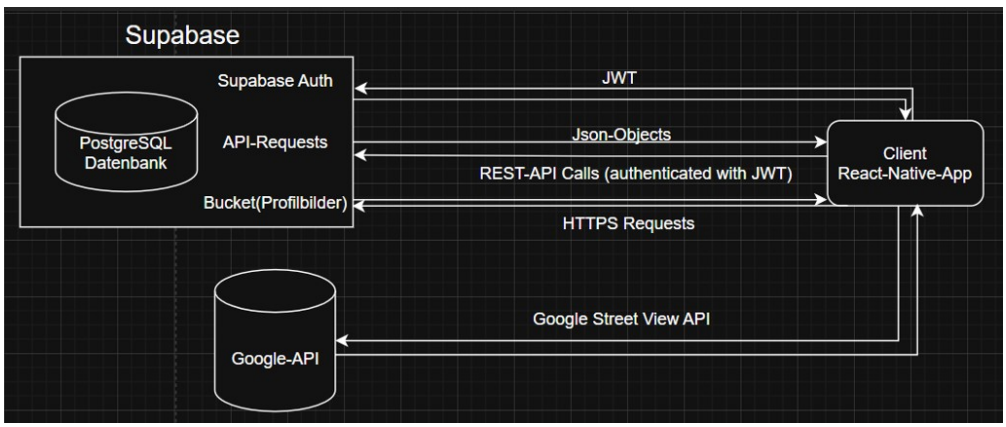

Abbildung 4.1.: Systemarchitektur der Anwendung WoSamma

4.2.2. Architektur der Datenbank

Abbildung 4.2.: Datenbankarchitektur der Anwendung WoSamma

Die Datenbankarchitektur der Anwendung *WoSamma* basiert auf einem relationalen Schema unter Verwendung von PostgreSQL (über Supabase). Im Zentrum des Modells steht die von Supabase bereitgestellte Tabelle `auth.users`, welche die Authentifizierungsdaten verwaltet. Darauf aufbauend werden benutzerspezifische Informationen in der Tabelle `profiles` gespeichert. Diese enthält unter anderem statistische Kennzahlen wie gespielte und gewonnene Spiele sowie Statusinformationen.

Die Spiellogik wird über mehrere miteinander verknüpfte Tabellen abgebildet. Die Tabelle `lobbies` verwaltet aktive Spielräume inklusive Host, Spielstatus und Rundeneinstellungen. Über die Zwischentabelle `lobby_players` wird die n:m-Beziehung zwischen Benutzern und Lobbys realisiert. Einladungen zu Spielrunden werden in `lobby_invites` gespeichert.

Einzelne Spielrunden werden in der Tabelle `user_rounds` dokumentiert. Dort werden unter anderem die tatsächlichen Koordinaten, die vom Benutzer gewählten Koordinaten, die Distanz sowie die erreichten Punkte gespeichert. Für den Einzelspielermodus existiert zusätzlich die Tabelle `user_daily_game`, während `daily_location` die täglich rotierenden Zielkoordinaten enthält.

Soziale Interaktionen werden durch eigene Tabellen strukturiert: `friend_requests` bildet Freundschaftsanfragen ab, `private_messages` sowie `news_messages` ermöglichen Kommunikationsfunktionen. Profilbilder werden über `profile_pictures` und `user_profile_pictures` verwaltet.

Administrative Prozesse sind ebenfalls integriert. Fehlermeldungen werden in `bug_reports` gespeichert, während die Tabelle `reports` Benutzer-Meldungen abbildet. Die Tabelle `config` dient der Speicherung globaler Konfigurationswerte.

Die Datenbank folgt dem Prinzip der Normalisierung, wodurch Redundanzen minimiert und Datenkonsistenz gewährleistet werden. Primär- und Fremdschlüssel stellen referenzielle Integrität sicher und ermöglichen eine klare Trennung zwischen Authentifizierungsdaten, Spiellogik, sozialen Funktionen und administrativen Prozessen. Durch diese modulare Struktur ist die Datenbank sowohl wartbar als auch skalierbar und kann bei steigenden Nutzerzahlen effizient erweitert werden.

4.2.3. Frontend-Implementierung & Planung

Die Planung und die dementsprechende Umsetzung des Frontends stellen einen wesentlichen Teil der Implementierung dar. Dieser Bereich ist sehr umfangreich und hatte eine hohe Priorität, da er die Kommunikation zwischen Benutzer und Applikation abbildet. Aufbauend auf den zuvor beschriebenen theoretischen Grundlagen zu User Experience, Informationsarchitektur und Interface-Design wurden zunächst die funktionalen Anforderungen an das Frontend definiert. Ziel war es, eine übersichtliche, intuitive und visuell konsistente Benutzeroberfläche zu schaffen, die sowohl im Einzel- als auch im Mehrspielerbetrieb eine einfache und verständliche Nutzung ermöglicht.

Als Grundlage der Planung dienten die Erstellung von Wireframes und einem Prototypen sowie die Ausarbeitung der Informationsarchitektur. Die eigentliche Implementierung orientiert sich an den meisten Stellen stark an dem zuvor erstellten Prototyp, da dieser bereits eine sehr genaue Vorstellung der finalen Benutzeroberfläche vermittelte.

4.2.3.1. Informationsarchitektur, Wireframes & Prototypen

Die entwickelte Informationsarchitektur orientiert sich an den zuvor beschriebenen theoretischen Grundlagen der Informationsarchitektur und setzt diese konsequent in eine klar strukturierte Navigationshierarchie um. Ausgehend von einem zentralen Einstiegspunkt (Home), welcher nach Login erreicht wird, werden die Inhalte logisch in thematisch zusammengehörige Bereiche wie Einzelspieler, Mehrspieler und Bestenliste gegliedert. Dabei wurde auf eine begrenzte Auswahl pro Navigationsebene sowie auf eine eindeutige Benennung der Menüpunkte geachtet, um die kognitive Belastung der Nutzer gering zu halten. Durch die klare Trennung von Kernfunktionen (z. B. Spielmodi) und unterstützenden Funktionen (z. B. Profil, Shop, Neuigkeiten) wird eine gute Orientierung gewährleistet.

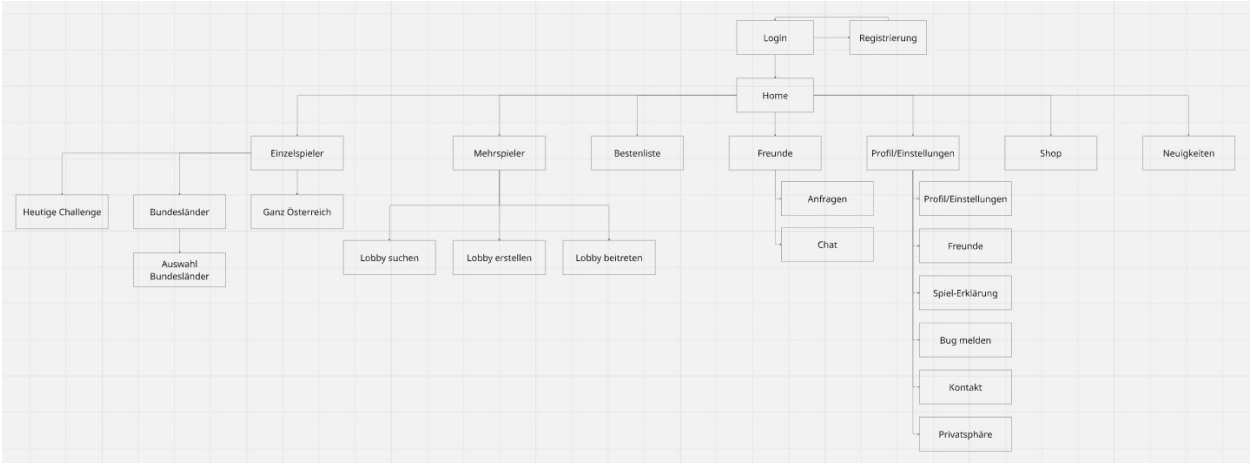

Abbildung 4.3.: Informationsarchitektur der Anwendung WoSamma

Das erstellte Wireframe diente als strukturelle Grundlage der Anwendung. Der Fokus lag auf der Anordnung der Inhalte, der grundsätzlichen Struktur der einzelnen Bereiche und der Führung der Nutzer durch das System. Visuelle Gestaltungselemente wie Farbkonzepte, Typografie oder detaillierte UI-Komponenten sind bewusst offen gelassen worden, um den Schwerpunkt auf Struktur, Hierarchie und Nutzerführung zu legen.

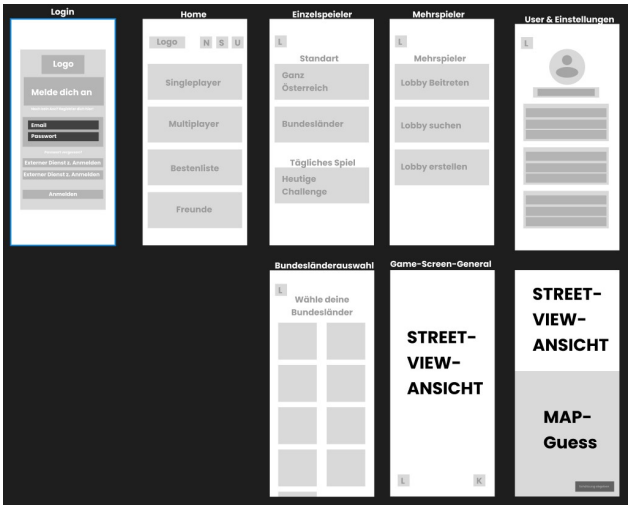

Abbildung 4.4.: Wireframes der Anwendung WoSamma

Die im vorherigen Kapitel beschriebenen Designkonzepte und Gestaltungsprinzipien wurden im Mockup gezielt und konsequent umgesetzt. Aufbauend auf den theoretischen Grundlagen zu modernem App-Design, Dark-Mode-Konzepten, Barrierefreiheit und klarer Informationshierarchie wurde das Mockup nach modernen und aktuellen Maßstäben entworfen. Die visuelle Gestaltung orientiert sich an einem durchgängigen Dark-Mode-Design mit bewusst gewählten Kontrasten, um eine gute Lesbarkeit unter unterschiedlichen Lichtbedingungen sicherzustellen.

Aspekte der Barrierefreiheit wurden im Mockup unter anderem durch ausreichend große Touch-Ziele, klare Beschriftungen, eindeutige Icons sowie eine logisch aufgebaute Navigations- und Fokusstruktur berücksichtigt. Die hierarchische Anordnung von Inhalten, wie in den Abschnitten zu Informationsarchitektur und Struktur beschrieben, spiegelt sich direkt in der Seitenaufteilung und Navigation des Mockups wider. Farben, Typografie und Abstände wurden so eingesetzt, dass visuelle Orientierung unterstützt und eine intuitive Nutzung ermöglicht wird.

Das Mockup dient damit nicht nur als visuelle Vorschau, sondern als konkrete Umsetzung der zuvor definierten UI und Designprinzipien und bildet die Grundlage für die technische Realisierung der Anwendung.

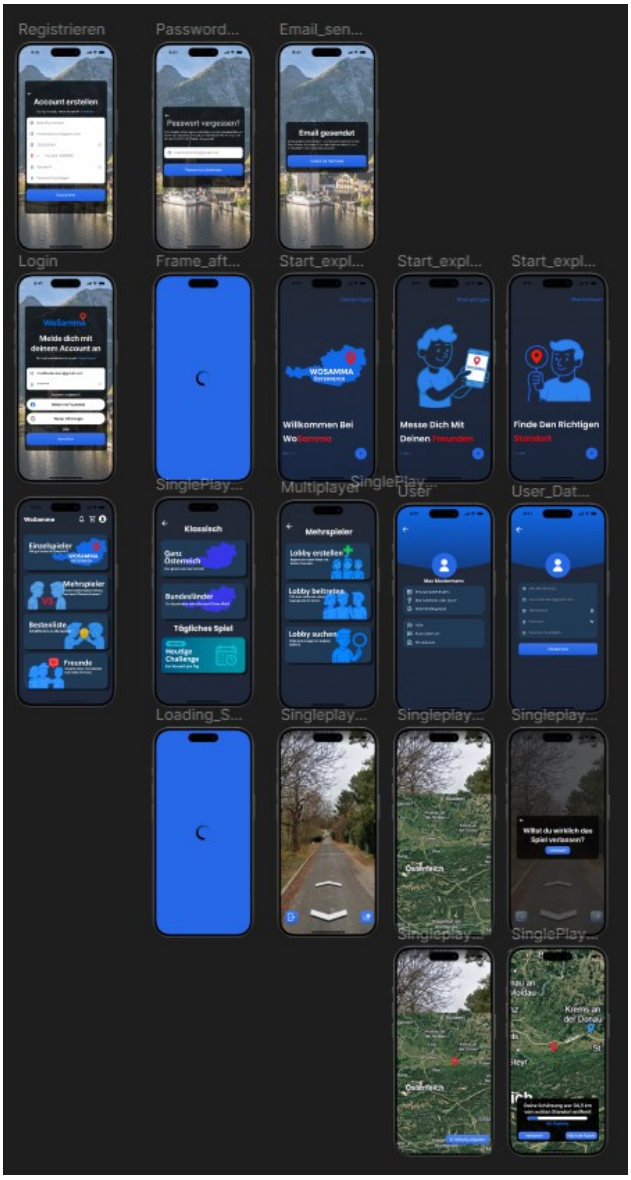

Abbildung 4.5.: Mockup der Anwendung WoSamma(Überblick)

4.2.3.2. Responsives Design

Das Responsive Design der Anwendung wurde durch den gezielten Einsatz flexibler Layout- und Größenkonzepte umgesetzt. Mithilfe von Flexbox (flex: 1) passen sich Container automatisch an den verfügbaren Bildschirmraum an, während prozentuale Breiten wie width: '95%' oder dynamische Werte auf Basis der Bildschirmbreite (windowWidth * 0.95) eine adaptive Skalierung auf unterschiedlichen Geräten ermöglichen. Zusätzlich sorgen Einschränkungen wie maxWidth, minHeight und maxHeight dafür, dass UI-Elemente auf großen wie kleinen Displays gut nutzbar bleiben. Ergänzend wurden feste Seitenverhältnisse (aspectRatio) für visuelle Konsistenz sowie KeyboardAvoidingView für ein geräteabhängiges, nutzerfreundliches Verhalten bei Texteingaben eingesetzt. Dadurch bleibt die Benutzeroberfläche über verschiedene Bildschirmgrößen und Nutzungsszenarien hinweg stabil, übersichtlich und bedienbar.

4.2.3.3. Spielansicht und Benutzerinteraktion

In den folgenden Abschnitten werden die wichtigsten Aspekte des UI sowie der Benutzerinteraktion beschrieben. Hierbei wird aktiv auf Code eingegangen und erläutert, wie dieser Funktioniert und welchen Zweck er erfüllt.

4.2.3.3.1. Pin-Platzierung

Die Platzierung des Pins auf der Karte ist die wohl zentralste Interaktionsmöglichkeit im Spiel. Ohne diese Funktion hätte das Spiel keinen Sinn, da der Spieler so seine Vermutung über den Standort abgibt. Diese Funktionalität wird hier rein anhand des Einzepspieler-Modus erklärt, da sich die reine Platzierungslogik im Mehrspieler-Modus bzw. in der täglichen Challenge nicht unterscheidet.

Zusätzlich zur reinen Platzierungslogik spielt die Benutzerinteraktion auf UI-Ebene eine zentrale Rolle. Die Karte registriert Klick- bzw. Touch-Ereignisse und setzt den Marker unmittelbar an der gewählten Position (e.latLng). Dadurch erhält der Spieler ein direktes visuelles Feedback ohne Verzögerung, was wesentlich für ein flüssiges und reaktionsschnelles Spielgefühl ist.

Um die Bedienung übersichtlich zu halten, ist stets nur ein aktiver Pin sichtbar. Wird eine neue Position gewählt, wird der zuvor gesetzte Marker entfernt und durch einen neuen ersetzt. Dieses sogenannte Single-Pin-Prinzip verhindert Mehrdeutigkeiten und erlaubt es dem Spieler, seine Schätzung beliebig oft zu korrigieren, ohne dass mehrere Marker die Karte überlagern. Gerade auf mobilen Endgeräten stellt dies eine intuitive und fehlerverzeihende Interaktion sicher.

Auch die initiale Kartenkonfiguration wurde bewusst auf die Spielmechanik abgestimmt. Die Startposition und der Zoom-Level sind so gewählt, dass Österreich vollständig sichtbar ist und der Spieler sich unmittelbar orientieren kann. Nicht benötigte UI-Elemente wie Street-View- oder Kartenmodus-Steuerungen wurden deaktiviert, um visuelle Ablenkungen zu reduzieren und den Fokus klar auf die Platzierung des Pins zu setzen.

Darüber hinaus sorgt die vollständige Ausnutzung des verfügbaren Bildschirmbereichs (height: 100%) dafür, dass die Karte sowohl auf mobilen Geräten als auch auf größeren Displays

konsistent dargestellt wird. Insgesamt trägt diese reduzierte, klar strukturierte UI dazu bei, die zentrale Spielhandlung, also das Abgeben einer Standortvermutung, effizient umzusetzen.

Der folgende Codeausschnitt zeigt, wie die Pin-Platzierung technisch umgesetzt wurde:

```
    <script>
      // Globale Variablen:
      // map: Referenz auf die Google Map
      // marker: Referenz auf den aktuell gesetzten Pin
      var map, marker = null;

      // Initialisiert die Google Map
      function initMap() {

        // Erstellen der Map mit Startposition Österreich
        map = new google.maps.Map(document.getElementById('map'), {
          center: { lat: 47.5162, lng: 14.5501 }, // Mittelpunkt Österreich
          zoom: 7,                                // Zoom-Level
          streetViewControl: false,               // UI-Elemente deaktiviert
          mapTypeControl: false,
          fullscreenControl: false,
          backgroundColor: '#181d23',
        });

        // Listener für Klicks auf die Karte
        map.addListener('click', function(e) {

          // Falls bereits ein Pin existiert, wird er entfernt
          // -> es gibt IMMER nur einen aktiven Guess-Pin
          if (marker) marker.setMap(null);

          // Neuer Marker wird an der Klickposition gesetzt
          marker = new google.maps.Marker({
            position: e.latLng,
            map: map
          });

          ...
        });
      }
      // Startet die Map nach dem Laden
      initMap();
    </script>
';
```

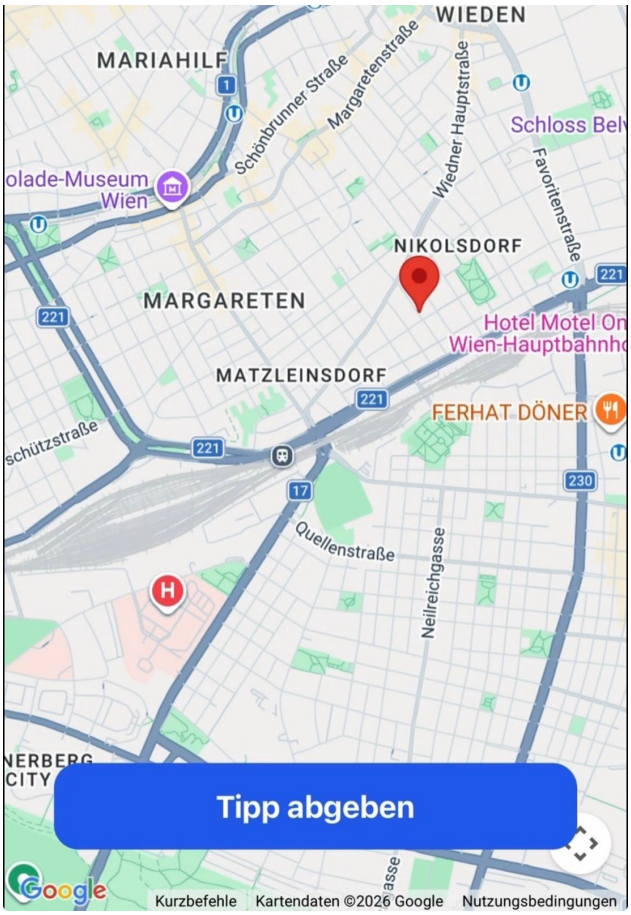

Abbildung 4.6.: platzierter Pin im Einzelspieler

4.2.3.3.2. Anzeige mehrerer Pins im Mehrspieler-Modus

Im Mehrspieler-Modus besteht die Herausforderung nicht nur in der Anzeige mehrerer Marker, sondern insbesondere in deren eindeutiger visueller Unterscheidbarkeit. Um eine klare Zuordnung zu ermöglichen, werden die Pins in unterschiedlichen Farben dargestellt und jeweils mit dem zugehörigen Benutzernamen versehen. Dadurch kann der Spieler auf einen Blick erkennen, welche Position welchem Mitspieler zugeordnet ist. Diese visuelle Differenzierung ist essenziell, um Verwechslungen zu vermeiden und die Vergleichbarkeit der Ergebnisse zu gewährleisten.

Darüber hinaus müssen alle Marker gleichzeitig auf derselben Karteninstanz gerendert werden. Die Karte dient somit als gemeinsame visuelle Referenzfläche, auf der sämtliche Schätzungen parallel angezeigt werden. Dabei ist darauf zu achten, dass sich Marker bei nah beieinanderliegenden Positionen nicht vollständig überdecken. Durch eine gezielte Steuerung der Rendering-Reihenfolge kann sichergestellt werden, dass relevante Marker, etwa der eigene Tipp, stets sichtbar bleiben.

Die konkrete visuelle Umsetzung der geschätzten sowie der korrekten Positionen wird im folgenden Abschnitt erläutert. An dieser Stelle wird zunächst beschrieben, wie die gleichzeitige Darstellung mehrerer Pins unterschiedlicher Spieler auf der Karte technisch realisiert werden kann:

1. Spieler setzt Pin auf Karte
2. Pin-Koordinaten werden im State gespeichert (`setPin`)
3. Bei Tipp abgeben → Speicherung in `user_rounds` (Datenbank):
   - `game_id` (Lobby-ID)
   - `user_id` (eigene ID)
   - `round_number`
   - `guess_lat`, `guess_lng`
   - `finished: true`
4. Echtzeit-Tracking (alle 2 Sekunden):
   - Prüft, wie viele Spieler bereits getippt haben
   - Zeigt Fortschritt: `{guessedCount}/{totalPlayers}`
5. Result-Map lädt alle Tipps:
   - Polling alle 2 Sekunden aus `user_rounds`
   - Lädt Usernames und Avatare
   - Zeigt alle Pins mit unterschiedlichen Farben und deren dazugehörigen Usernamen, der Eigene wird mit höherem zIndex angezeigt für bessere Sichtbarkeit
   - Zeigt Linien zur Lösung und Distanz-Labels (genauere Beschreibung im nächsten Abschnitt)

Der folgende Codeausschnitt zeigt, wie man es nun schafft all die gespeicheerten Pins aus der Datenbank auf der Karte anzuzeigen:

```
/**
 * Erstellt den HTML-/JavaScript-Code für die Ergebnis-Karte im Mehrspieler-Modus.
 * In diesem Abschnitt liegt der Fokus ausschließlich auf der Darstellung
 * mehrerer Pins unterschiedlicher Spieler auf der Karte.
 *
 * Die ausgelesenen Tipps sind als Array namens "guesses" vorhanden, diese werden vorher aus der Datenbank gelesen und aufbereitet.
 *
 * Die Visualisierung der Verbindungslinien zwischen Tipp und korrekter Lösung
 * sowie die Distanzberechnung werden bewusst ausgelassen und im nächsten
 * Abschnitt separat erläutert.
 */
function getResultMapHTML({ solutionLat, solutionLng, guesses, apiKey, selfUserId }: any) {

  // Vordefinierte Google-Maps-Standard-Icons in unterschiedlichen Farben.
  // Die Farben rotieren, sodass mehrere Spieler gleichzeitig klar unterscheidbar sind.
  const pinIcons = [
    'http://maps.google.com/mapfiles/ms/icons/red-dot.png',
    'http://maps.google.com/mapfiles/ms/icons/blue-dot.png',
```

```
    'http://maps.google.com/mapfiles/ms/icons/green-dot.png',
    'http://maps.google.com/mapfiles/ms/icons/yellow-dot.png',
    'http://maps.google.com/mapfiles/ms/icons/purple-dot.png',
    'http://maps.google.com/mapfiles/ms/icons/orange-dot.png',
    'http://maps.google.com/mapfiles/ms/icons/pink-dot.png',
    'http://maps.google.com/mapfiles/ms/icons/cyan-dot.png',
];

//@ts-ignore
// Für jeden gespeicherten Tipp wird JavaScript-Code erzeugt,
// der einen eigenen Marker inklusive Username-Overlay erstellt.
const markersJS = guesses.map((g, i) => `

  // === Marker (Pin) für einen Spieler ===
  // Jeder Marker repräsentiert einen gespeicherten Tipp aus der Datenbank
  var marker = new google.maps.Marker({
    position: { lat: ${g.lat}, lng: ${g.lng} }, // Pin-Koordinaten des Spielers
    map: map,                                   // Anzeige auf der Ergebnis-Karte

    // Farbiger Pin zur visuellen Unterscheidung der Spieler
    icon: '${pinIcons[i]}',

    // Username als Tooltip (z.B. sichtbar bei Hover am Desktop)
    title: '${g.username}',

    // Der eigene Pin wird priorisiert gerendert,
    // damit er bei Überlagerungen stets sichtbar bleibt
    zIndex: ...
  });

  // === Username-Overlay über dem Pin ===
  // Der Name des Spielers wird direkt über dem Marker angezeigt,
  // um eine eindeutige Zuordnung der Tipps zu ermöglichen
  (function() {
    var nameDiv = document.createElement('div');

    // UI-Styling für gute Lesbarkeit auf der dunklen Kartenansicht
    nameDiv.style.fontSize = '13px';
    nameDiv.style.fontWeight = 'bold';
    nameDiv.style.color = '#fff';
    nameDiv.style.background = 'rgba(67,160,71,0.85)';
    nameDiv.style.padding = '2px 8px';
    nameDiv.style.borderRadius = '8px';
    nameDiv.style.textShadow = '0 0 2px #000, 0 0 1px #000';

    nameDiv.innerText = '${g.username}';

    // OverlayView ermöglicht das Platzieren von HTML-Elementen über der Map
    var nameOverlay = new google.maps.OverlayView();

    // Hinzufügen des Namens zur Overlay-Ebene der Karte
    nameOverlay.onAdd = function() {
```

```
71          var panes = this.getPanes();
72          panes.overlayImage.appendChild(nameDiv);
73        };
74
75        // Positionierung des Overlays relativ zum Marker
76        nameOverlay.draw = function() {
77          var projection = this.getProjection();
78          var pos = projection.fromLatLngToDivPixel(
79            new google.maps.LatLng(${g.lat}, ${g.lng})
80          );
81          nameDiv.style.left = pos.x + 'px';
82          nameDiv.style.top = (pos.y - 28) + 'px'; // leicht oberhalb des Pins
83          nameDiv.style.position = 'absolute';
84          nameDiv.style.transform = 'translate(-50%, -100%)';
85        };
86
87        nameOverlay.setMap(map);
88      })();
89
90      /*
91        * Hinweis:
92        * Die Darstellung der Verbindungslinien zwischen den getippten
      Positionen
93        * und der korrekten Lösung sowie die zugehörige Distanzberechnung
94        * werden im nächsten Abschnitt detailliert beschrieben und sind
95        * hier bewusst nicht enthalten.
96        */
97
98    `).join('\n');
99  }
```

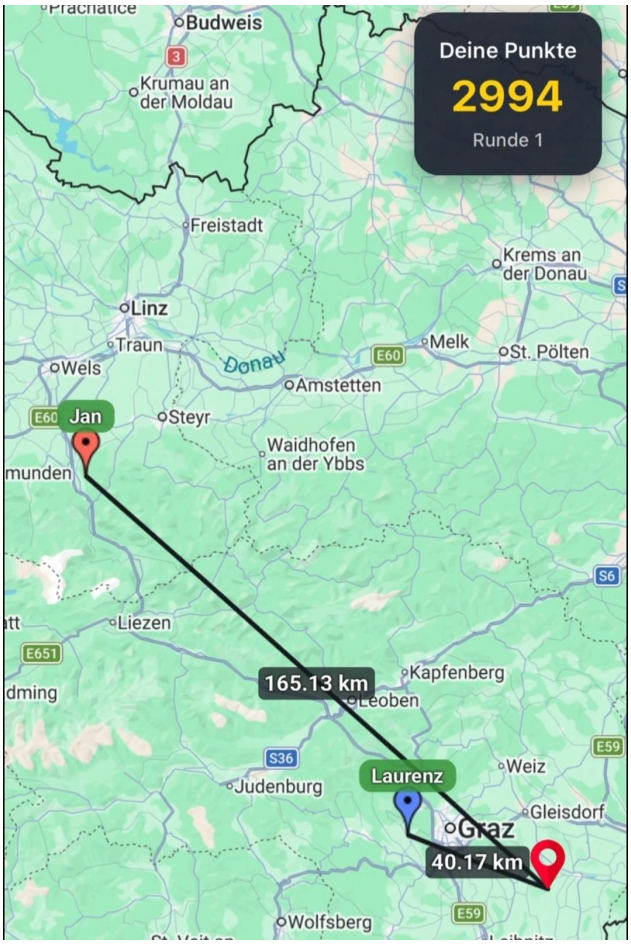

Abbildung 4.7.: Anzeige mehrerer Pins im Mehrspieler

4.2.3.3.3. Visualisierung der geschätzten und korrekten Position
Für die Ergebnisdarstellung ist nicht nur das Anzeigen beider Positionen, also der korrekten Position und der geschätzen des Spielers entscheidend, sondern vor allem deren visuelle Verknüpfung. Die Linie zwischen getippter und tatsächlicher Position fungiert dabei als zentrales Orientierungselement. Sie macht die räumliche Abweichung unmittelbar sichtbar und ergänzt die numerische Distanzangabe um eine intuitive, grafische Komponente. Dadurch wird das Ergebnis nicht nur berechnet, sondern auch visuell erfahrbar gemacht.

Die Linien werden als eigenständige Overlay-Elemente auf der bestehenden Karteninstanz gerendert. Dabei greifen sie auf dieselben geografischen Koordinaten zurück wie die Marker und passen sich automatisch an Zoomstufe und Kartenausschnitt an. Unabhängig vom gewählten Spielmodus bleiben grundlegende Eigenschaften wie Linienstärke, Transparenz und Verlauf konsistent, um eine einheitliche visuelle Struktur sicherzustellen. Lediglich die Farbgebung kann variieren, beispielsweise zur besseren Unterscheidung mehrerer Spieler im Mehrspieler-Modus.

Zusätzlich wird die Linie so gestaltet, dass sie sich klar vom Kartenhintergrund abhebt, ohne andere UI-Elemente zu überdecken. Diese bewusste visuelle Gewichtung stellt sicher, dass

das Ergebnis schnell erfasst werden kann, ohne die Übersichtlichkeit der gesamten Karte zu beeinträchtigen.

Egal welchen Modus der Spieler spielt, die Linien eigenschaften sind bis auf die Farbe immer gleich und werden in diesem Code-Block definiert:

```
var line = new google.maps.Polyline({
  path: [ { lat: ${solutionLat}, lng: ${solutionLng} }, { lat: ${guessLat}, lng: ${guessLng} } ], //Weg der gezeichnet werden soll zwischen Guess des Spielers und der Lösung
  geodesic: true,        // Kurvenlinie entlang der Erdoberfläche
  strokeColor: '#2563eb', // Blaue Linie in allen Einzelspieler-Modi, im Mehrspieler ist diese Linie Schwarz/Dunkelgrau
  strokeOpacity: 1.0,   // Volle Deckkraft
  strokeWeight: 3,       // Linienstärke 3px
  map: map
});
```

Ein weiterer wichtiger Aspekt ist die Anzeige der tatsächlichen Distanz in Kilometern zwischen dem gesetzten Pin und der korrekten Position. Im Einzelspieler-Modus bzw. in der Täglichen Challenge wird die Distanz nicht direkt über der Verbindungsinie, sondern unten am Screen angezeigt:

```
<View style={styles.resultBox}>
  {/* Textuelles Feedback für den Spieler:
      Zeigt direkt an, wie weit die Schätzung vom tatsächlichen Standort entfernt war */}
  <Text style={styles.resultText}>
    Deine Schätzung war ... km vom echten Standort entfernt
  </Text>

  {/* Fortschrittsbalken zur visuellen Darstellung der erreichten Punkte.
      Der Balken skaliert relativ zur maximal erreichbaren Punktzahl (5000). */}
  <View style={styles.progressBarBg}>
    <View
      style={[
        styles.progressBar,
        {
          // Begrenzung der Breite auf 0-100 %, um Layout-Probleme zu vermeiden
          width: '...',
        },
      ]}
    />
  </View>

  {/* Anzeige der erreichten Punktzahl als numerischer Wert */}
  <Text style={styles.points}>
    ... Punkte
  </Text>

  {/* Button-Leiste für weitere Aktionen nach Abschluss der Runde */}
  <View style={styles.buttonRow}>
```

```
29
30      {/* Button zum Verlassen der aktuellen Lobby / Runde.
31           Öffnet ein Bestätigungs-Modal, um unbeabsichtigtes Verlassen zu
        verhindern */}
32      <TouchableOpacity
33        style={styles.leaveBtn}
34        onPress={...}
35      >
36        <Text style={styles.leaveText}>Verlassen</Text>
37      </TouchableOpacity>
38
39      {/* Button zum Starten der nächsten Runde.
40           Je nach Spielmodus (z.B. Bundesländer-Modus oder Österreich gesamt
        )
41           wird der entsprechende Screen geladen */}
42      <TouchableOpacity
43        style={styles.nextBtn}
44        onPress={...}
45      >
46        <Text style={styles.nextText}>Nächste Runde</Text>
47      </TouchableOpacity>
48    </View>
49  </View>
```

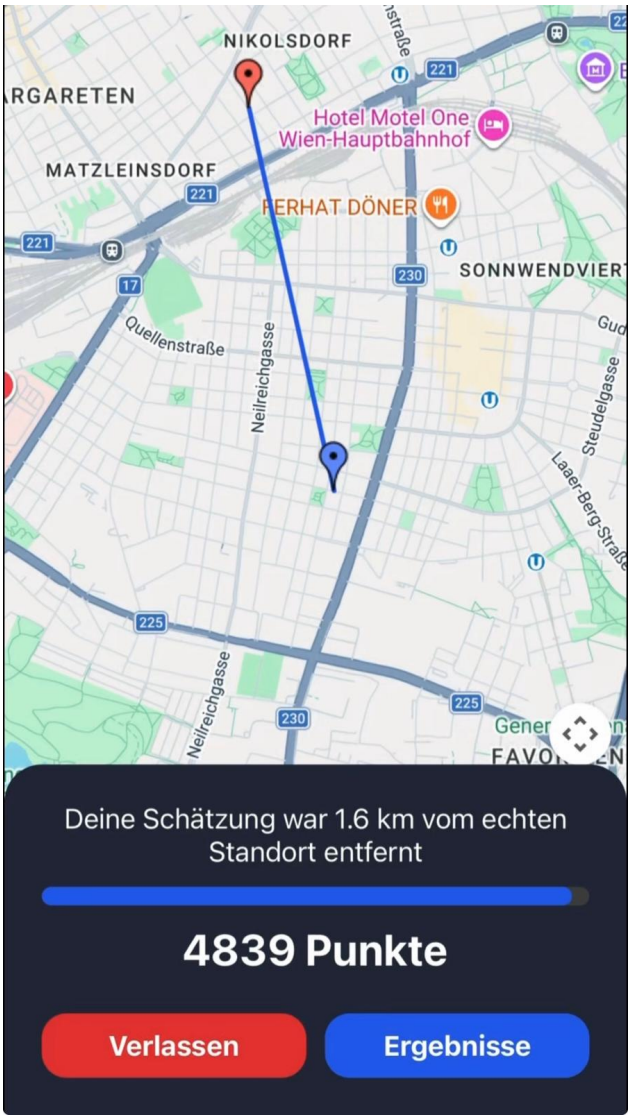

Abbildung 4.8.: Entfernungsanzeige im Einzelspieler

Im Gegensatz dazu, wird im Mehrspieler-Modus die Distanz zwischen der getippten Position und der korrekten Lösung direkt auf der Verbindungslinie angezeigt. Dadurch können alle Spieler nicht nur ihre eigene, sondern auch die Abweichungen der Mitspieler nachvollziehen und miteinander vergleichen. Die Umsetzung dieser Visualisierung erfolgt mithilfe des folgenden Codeausschnitts:

```
var labelDiv = document.createElement('div');

  // UI-Styling: gut lesbar auf dunkler Karte / über der Linie
  labelDiv.style.fontSize = '14px';
  labelDiv.style.fontWeight = 'bold';
  labelDiv.style.color = '#fff';
  labelDiv.style.textShadow = '0 0 2px #000, 0 0 1px #000';
  labelDiv.style.padding = '2px 4px';
  labelDiv.style.borderRadius = '4px';
```

```
10  labelDiv.style.background = 'rgba(24,29,35,0.7)';
11
12  // Text setzen (auf 2 Nachkommastellen gerundet)
13  labelDiv.innerText = '... km';
14
15  // OverlayView: erlaubt HTML-Elemente pixelgenau über der Google Map zu
      platzieren
16  var label = new google.maps.OverlayView();
17
18  // Overlay in die passende Map-Ebene einhängen
19  label.onAdd = function() {
20    var panes = this.getPanes();
21    panes.overlayImage.appendChild(labelDiv);
22  };
23
24  // draw(): wird von Google Maps aufgerufen, sobald Projektion/Zoom
      bekannt ist
25  // Hier wird Lat/Lng (Mittelpunkt) in Pixelposition umgerechnet
26  label.draw = function() {
27    var projection = this.getProjection();
28    var pos = projection.fromLatLngToDivPixel(new google.maps.LatLng(midLat, midLng));
29
30    // Label mittig auf die Linie setzen
31    labelDiv.style.left = pos.x + 'px';
32    labelDiv.style.top = pos.y + 'px';
33    labelDiv.style.position = 'absolute';
34    labelDiv.style.transform = 'translate(-50%, -50%)';
35  };
36
37  // Overlay aktivieren, Label wird angezeigt
38  label.setMap(map);
```

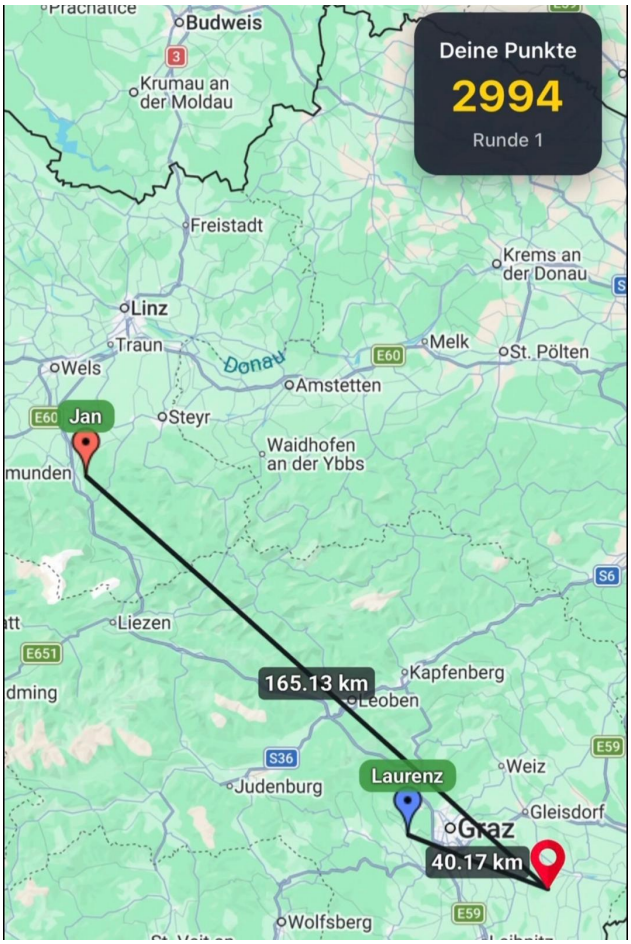

Abbildung 4.9.: Anzeige der Entfernung im Mehrspielermodus

4.2.3.3.4. Zoom Animationen

Um dem Spieler nach Abschluss einer Runde ein besseres visuelles Feedback zu geben, wird in den Einzelspieler-Modi eine Zoom-Animation verwendet. Dabei startet die Ergebnis-Karte bewusst mit einem sehr weit herausgezoomten Kartenbild (Zoom-Level 3) und zoomt anschließend schrittweise auf ein zuvor berechnetes, optimales Zoom-Level "targetZoom" heran. Diese Animation sorgt dafür, dass die Ergebnisansicht dynamischer wirkt und die Aufmerksamkeit des Spielers auf die relevante Region gelenkt wird. Die Zoom-Animation erfüllt dabei nicht nur einen ästhetischen Zweck, sondern unterstützt gezielt die Benutzerführung. Durch das initiale Herauszoomen erhält der Spieler zunächst eine räumliche Gesamtübersicht, wodurch die relative Lage von getippter und tatsächlicher Position im größeren geografischen Kontext sichtbar wird. Das anschließende schrittweise Heranzoomen fokussiert die Aufmerksamkeit dann kontrolliert auf den relevanten Kartenausschnitt. Dieser Ablauf erzeugt eine klare visuelle Dramaturgie und verhindert, dass der Nutzer abrupt in eine stark vergrößerte Detailansicht versetzt wird.

Technisch wird die Animation direkt auf der bestehenden Karteninstanz durchgeführt, indem das Zoom-Level in definierten Intervallen angepasst wird, bis das zuvor berechnete Zielniveau erreicht ist. Die Karte rendert jede Zwischenstufe automatisch neu, wodurch eine flüssige

Übergangsbewegung entsteht. Dabei bleibt die Kartenmitte konstant auf die relevante Region ausgerichtet, sodass keine zusätzliche Verschiebung (Pan-Bewegung) erforderlich ist. Der folgende Codeausschnitt zeigt die technische Umsetzung dieser Zoom-Animation:

```
// Zoom-Animation (Einzelspieler)
// Startet weit herausgezoomt und zoomt schrittweise bis zum optimalen Zoom-Level (targetZoom)
var currentZoom = 3;

// Intervall steuert die Zoom-Schritte -> kleine Inkremente sorgen für eine flüssige Animation
var zoomInterval = setInterval(function() {

  // Solange der aktuelle Zoom kleiner als das Ziel ist, wird in kleinen Schritten gezoomt
  if (currentZoom < ${targetZoom}) {
    currentZoom += 0.2;      // kleiner Zoom-Schritt für "smooth" Effekt
    map.setZoom(currentZoom);
  } else {
    // Endzustand: Zielzoom setzen und Intervall beenden
    map.setZoom(${targetZoom});
    clearInterval(zoomInterval);
  }

}, 40); // Aktualisierung alle 40ms -> ergibt eine sichtbare, aber nicht hektische Animation
```

Im Mehrspieler-Modus wird hingegen bewusst auf eine Zoom-Animation verzichtet. Stattdessen wird die Karte mit einem fixen Zoom-Level (Zoom 7) initialisiert und auf die korrekte Lösung zentriert. Der Grund dafür ist, dass im Mehrspieler-Modus mehrere Pins gleichzeitig sichtbar sein müssen. Ein fixer Zoom-Level bietet hier eine konsistente und faire Darstellung, da alle Spieler dieselbe Kartenansicht erhalten und nicht durch individuelle Zoom-Animationen oder unterschiedliche Bildausschnitte beeinflusst werden.

Der folgende Codeausschnitt zeigt die Initialisierung der Ergebnis-Karte im Mehrspieler-Modus:

```
function initMap() {
  // Mehrspieler: fester Zoom und Zentrierung auf die korrekte Position,
  // damit möglichst viele Spieler-Pins gleichzeitig sichtbar bleiben
  var map = new google.maps.Map(document.getElementById('map'), {
    center: { lat: ${solutionLat}, lng: ${solutionLng} }, // Fokus auf der Lösung
    zoom: 7,                                              // fixer Zoom-Level ohne Animation

    // Deaktivierte Controls, um UI-Überladung zu vermeiden (WebView-optimiert)
    streetViewControl: false,
    mapTypeControl: false,
    fullscreenControl: false,
    backgroundColor: '#181d23',
  });
}
```

4.2.3.3.5. Darstellung von roten Begrenzungslinien der Bundesländer
Um dem Spieler die Orientierung im Bundesländer-Modus zu erleichtern, wurde es als notwendig erachtet, die ausgewählten Regionen direkt auf der Karte hervorzuheben. In der fertigen Applikation geschieht dies durch rote Begrenzungslinien, welche die jeweiligen Bundesländer umrahmen. Diese Visualisierung ermöglicht es auch Spielern ohne detaillierte geografische Kenntnisse, ihre Schätzung deutlich zu verbessern. Aber auch für Spieler, die grundsätzlich wissen, wo sich die einzelnen Bundesländer befinden, erweist sich diese Funktion als hilfreich, da die exakten Grenzen eines Bundeslandes meist nicht positionsgenau bekannt sind.

Technisch werden die Begrenzungslinien als separate Polygon-Overlays auf der bestehenden Karteninstanz dargestellt. Die geografischen Koordinaten der jeweiligen Bundesländer werden dabei als Punktlisten definiert und an die Karten-API übergeben, sodass geschlossene Flächen entstehen, die exakt den Verwaltungsgrenzen entsprechen. Durch eine klar definierte Linienfarbe sowie eine erhöhte Strichstärke heben sich die Umrisse deutlich vom Kartenhintergrund ab, ohne jedoch andere Spielinformationen zu überdecken.

Die Darstellung ist dabei rein visuell unterstützend und beeinflusst die eigentliche Spielmechanik nicht. Sie dient ausschließlich der Orientierung innerhalb des gewählten Modus. Da die Polygone direkt auf der Karte gerendert werden, passen sie sich automatisch an Zoomstufe und Kartenausschnitt an. Dadurch bleiben die Grenzen sowohl in der Gesamtansicht als auch in detaillierten Zoomstufen konsistent und klar erkennbar. Im folgenden Codeabschnitt wird die technische Umsetzung dieser Begrenzungslinien dargestellt:

```
/**
 * Zeichnet die Bundesland-Grenzen als rote Linien (Polyline) auf die Google Map.
 * Wichtig: Es werden nur die zuvor ausgewählten Bundesländer visualisiert.
 */
function drawBundeslandBorders(map: google.maps.Map, polygons: number[][][]) {
  // polygons = Array von Polygonen
  // Jedes Polygon besteht aus Koordinatenpaaren: [ [lat, lng], [lat, lng], ... ]
  polygons.forEach(function (poly) {
    // Google Maps erwartet ein Array aus Objekten: { lat: ..., lng: ... }
    var path = poly.map(function (coord) {
      return { lat: coord[0], lng: coord[1] };
    });

    // Polyline = Linie entlang eines Pfades
    // -> wird hier verwendet, um die Grenzen eines Bundeslandes als "Umrandung" darzustellen
    new google.maps.Polyline({
      path: path,           // Koordinaten der Grenze
      geodesic: true,       // Linien folgen der Erdkrümmung (optisch besser)
      strokeColor: '#e53935', // Rot = klare Hervorhebung der Region
      strokeOpacity: 0.8,   // leicht transparent, damit Map darunter sichtbar bleibt
      strokeWeight: 3,      // Linienstärke
      map: map              // Linie wird auf dieser Map gerendert
    });
```

```
  });
}
```

Abbildung 4.10.: Begrenzungslinien für Niesterösterreich und Wien

Die Datenstruktur "bundeslandPolygone" enthält die geografischen Koordinaten, welche zur Darstellung der Umrandungen der einzelnen Bundesländer auf der Karte verwendet werden. Diese Koordinaten wurden zuvor definiert und werden, abhängig von der Auswahl des Nutzers, an den nächsten Screen übergeben und dort, wie im vorherigen Codeabschnitt zu sehen ist, in der Variable "polygons" gespeichert. Im folgenden Codeausschnitt ist beispielhaft die Koordinatenliste von Niederösterreich dargestellt:

```
const bundeslandPolygone: Record<string, [number, number][]> = {
  NO: [
    [47.95043408875506, 14.698762389554815],
    [48.04048385766521, 14.493588517608657],
    [48.0878393614031, 14.497187229454624],
    [48.12096875315342, 14.532637250068547],
    ...
  ],...}
```

4.2.3.3.6. Kompass in der Spielansicht

Der Kompass in der Spielansicht ist ein wichtiges UI-Feature, weil er dem Spieler jederzeit Orientierung gibt und die Street-View-Perspektive lesbarer macht. Da sich die Kamera in Google Street View frei drehen lässt, liefert der Kompass eine direkte Rückmeldung über die Blickrichtung (Heading) und verhindert, dass sich Nutzer beim Erkunden oder beim Platzieren eines Tipps verlaufen. Zusätzlich erhöht das Feature die Immersion, weil es die reale Blickrichtung aus Street View in ein klares, spieltypisches Overlay übersetzt, ein

kleines Detail, das sich für den Nutzer aber als hilfreich darstellt und das Gameplay spürbar verbessert. Hier kannst du ebenfalls noch einen technisch orientierten Zusatz ergänzen, der die UI-Implementierung etwas greifbarer macht:

Technisch wird der Kompass als fest positioniertes Overlay über der Street-View-Ansicht dargestellt. Er ist unabhängig vom eigentlichen Karten- bzw. Street-View-Element im Layout verankert und bleibt somit stets sichtbar, auch wenn sich die Perspektive dynamisch ändert. Die Rotation des Kompasszeigers erfolgt synchron zur aktuellen Blickrichtung der Street-View-Kamera, indem der jeweilige Heading-Wert ausgelesen und in eine entsprechende CSS-Transformation übersetzt wird. Dadurch entsteht eine direkte visuelle Kopplung zwischen Kamerabewegung und UI-Element.

Besonders wichtig ist dabei die kontinuierliche Aktualisierung bei jeder Richtungsänderung. Sobald der Spieler die Ansicht dreht, passt sich der Kompass in Echtzeit an. Diese unmittelbare Rückmeldung sorgt für Konsistenz zwischen Interaktion und Darstellung und stärkt die räumliche Orientierung. Gleichzeitig bleibt das Design bewusst reduziert, sodass der Kompass informativ wirkt, ohne die eigentliche Spielansicht zu überlagern oder visuell zu dominieren.

```
/**
 * CompassOverlay
 * - heading kommt aus StreetView (pov.heading)
 * - der Kompass wird als Overlay gezeichnet (SVG)
 * - die Nadel (bzw. das ganze Kompass-Element) wird via Animated Rotation gedreht
 */
export function CompassOverlay(props: { heading: number }) {
  const { heading } = props;

  // Animated Wert, der die aktuelle Rotation (in Grad) hält
  const rotation = useRef(new Animated.Value(0)).current;

  // Wir merken uns den zuletzt dargestellten Winkel, um den kürzesten Drehweg zu finden
  const lastHeadingRef = useRef(0);

  useEffect(() => {
    // StreetView liefert heading als Zahl, wir normalisieren auf 0...360
    const next = normalizeAngle(heading);
    const prev = normalizeAngle(lastHeadingRef.current);

    const shortestNext = shortestAngleTarget(prev, next);

    // Animation zur neuen Rotation
    Animated.timing(rotation, {
      toValue: shortestNext,
      duration: 120,
      useNativeDriver: true,  // Rotation kann nativ laufen, ergibt angenehme Animation
    }).start();

    // lastHeading updaten (wichtig fürs nächste Update)
    lastHeadingRef.current = shortestNext;
  }, [heading, rotation]);
```

```
33
34    /**
35      * Animated.Value -> String für transform rotate
36      * Wir geben "deg" aus, weil RN rotate einen String erwartet.
37      */
38    const rotate = rotation.interpolate({
39      inputRange: [-3600, 3600], // großzügig, weil shortestNext auch mal über
        360 hinausgehen kann
40      outputRange: ["-3600deg", "3600deg"],
41    });
42
43    return (
44      <View style={styles.compassOverlay} pointerEvents="none">
45        <View style={styles.compassCircle}>
46          {/* Wir drehen das ganze SVG, damit die Nadel "mitgeht" */}
47          <Animated.View style={{ transform: [{ rotate }] }}>
48            <Svg width={56} height={56} viewBox="0 0 56 56">
49              {/* Außenring */}
50              <Circle
51                ...
52              />
53
54              {/* Rote Nadel: Norden (zeigt im SVG "nach oben") */}
55              <Line
56                ...
57              />
58
59              {/* Weiße Nadel: Süden (zeigt im SVG "nach unten") */}
60              <Line
61                ...
62              />
63
64              {/* Labels */}
65              <SvgText
66                ...
67              >
68                N /* N für Norden */
69              </SvgText>
70
71              <SvgText
72                ...
73              >
74                S /* S für Süden */
75              </SvgText>
76            </Svg>
77          </Animated.View>
78        </View>
79      </View>
80    );
81  }
82
83  /**
84    * Winkel in den Bereich 0..360 bringen
85    */
```

```
86 function normalizeAngle(deg: number) {
87   const x = deg % 360;
88   return x < 0 ? x + 360 : x;
89 }
90
91 /**
92  * Gibt ein Ziel zurück, das vom aktuellen Winkel aus den kürzesten
      Drehweg nimmt.
93  * Dadurch läuft die Animation glatt und ohne Sprünge.
94  */
95 function shortestAngleTarget(prev: number, next: number) {
96   let delta = next - prev;
97
98   if (delta > 180) delta -= 360;
99   if (delta < -180) delta += 360;
100
101   return prev + delta;
102 }
103
104 const styles = StyleSheet.create({
105   // Positionierung als Overlay ( oben links)
106   compassOverlay: {
107     ...
108   },
109   // Kreis-Container (Hintergrund + Rundung)
110   compassCircle: {
111     ...
112   },
113 });
```

Abbildung 4.11.: Kompass in der Spieleansicht

4.2.4. Entity-Relationship-Diagramm

Das folgende Entity-Relationship-Diagramm (ERD) zeigt die wichtigsten Datenbanktabellen und deren Beziehungen für die Anwendung *WoSamma*. Es umfasst die zentralen Entitäten wie `users`, `profiles`, `lobbies`, `rounds` und `user_rounds`, sowie die Verknüpfungen zwischen diesen Tabellen, um die komplexen Interaktionen im Mehrspieler-Modus zu unterstützen. Das ERD dient als Grundlage für das Verständnis der Datenstruktur und der Geschäftslogik, die in der Backend-Implementierung mit Supabase realisiert wird.

BILDDDDDD

4.2.5. Backend-Anbindung mit Supabase

Das Backend der Anwendung *WoSamma* basiert auf Supabase und stellt zentrale Server-Funktionalitäten als *Backend-as-a-Service* bereit. Dadurch werden Kernaufgaben wie Authentifizierung, Datenpersistenz sowie Echtzeit-Kommunikation über eine konsistente API abstrahiert. Die mobile Anwendung kommuniziert über das Supabase-SDK mit der PostgreSQL-Datenbank und nutzt dabei eine tokenbasierte Zugriffskontrolle. Schreib- und Leseoperationen erfolgen ausschließlich innerhalb der durch Supabase bereitgestellten Sicherheitsmechanismen (z. B. JWT-basierte Sessions und Row Level Security), wodurch eine klare Trennung zwischen Client-Logik und serverseitiger Datenhaltung gewährleistet wird.

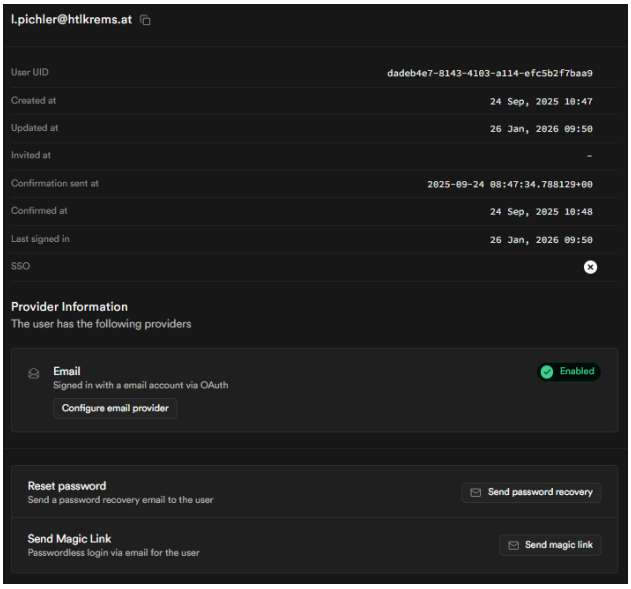

Abbildung 4.12.: Benutzerverwaltung und Authentifizierungsübersicht in Supabase

Abbildung 4.12 zeigt die Authentifizierungsoberfläche von Supabase für einen registrierten Benutzer. Dargestellt sind zentrale Metadaten wie die eindeutige Benutzer-ID, Zeitstempel für Registrierung, Bestätigung und letzte Anmeldung sowie der verwendete Authentifizierungsprovider. Zusätzlich bietet die Oberfläche administrative Funktionen wie Passwort-Zurücksetzung oder Magic-Link-Anmeldung.

Diese Ansicht verdeutlicht, wie Supabase die Benutzerverwaltung zentral kapselt und sicherheitsrelevante Prozesse serverseitig organisiert. Die Anwendung greift ausschließlich über autorisierte Sessions auf diese Daten zu, wodurch eine konsistente und sichere Authentifizierung gewährleistet wird.

4.2.5.1. Echtzeit-Kommunikation über Supabase Channels

Die Übertragung von Echtzeitdaten erfolgt über sogenannte *Channels*, die als persistente Kommunikationskanäle zwischen Client und Backend fungieren. Diese Kanäle können von der Anwendung abonniert (*subscribed*) werden, um Datenbankereignisse oder benutzerdefinierte Nachrichten unmittelbar zu empfangen.

Durch dieses Publish-Subscribe-Modell wird eine latenzarme und skalierbare Synchronisation zwischen mehreren Benutzern ermöglicht. Dies ist insbesondere für interaktive Funktionen wie Live-Einladungen, Mehrspieler-Lobbys und Chat-Kommunikation von zentraler Bedeutung, da Zustandsänderungen ohne explizite Polling-Mechanismen unmittelbar in der Benutzeroberfläche reflektiert werden.

Ein typisches Beispiel für die Struktur einer über einen Channel übertragenen Nachricht ist in Listing 4.1 dargestellt. Dieses JSON zeigt vereinfacht, wie Benutzerdaten oder Statusänderungen zwischen Client und Backend ausgetauscht werden.

Quellcode 4.1: Beispielhafte Echtzeit-Nachricht über einen Supabase Channel

```
{
  "eventType": "UPDATE",
  "table": "users",
  "new": {
    "id": "dadeb4e7-8143-4103-a114-efc5b2f7baa9",
    "username": "Kliv",
    "email": "l.pichler@htlkrems.at",
    "status": "online",
    "updated_at": "2026-01-26T08:50:08Z"
  }
}
```

Das Listing verdeutlicht, dass neben dem Ereignistyp auch die betroffenen Datensätze übertragen werden. Die Anwendung kann diese Informationen unmittelbar verarbeiten und die Benutzeroberfläche entsprechend aktualisieren.

Quellcode 4.2: Realtime-Subscription zur Synchronisation von Freundschaftsanfragen

```
// Initialisierung einer Realtime-Subscription für die Tabelle "friend_requests"
// Zweck dieses Codes ist es, die Freundesliste und die Freundschaftsanfragen
// in der Benutzeroberfläche in Echtzeit zu synchronisieren. Dadurch werden
// Änderungen wie das Hinzufügen, Annehmen oder Entfernen von Freunden sofort
// sichtbar, ohne dass die Ansicht manuell neu geladen werden muss.
subscription = supabase.channel('friend-requests-realtime')
  .on(
    'postgres_changes',
    {
      // Reaktion auf alle relevanten Datenbankereignisse (INSERT, UPDATE, DELETE),
      // um jede Statusänderung einer Freundschaftsanfrage zu erfassen
      event: '*',

      // Angabe des Datenbankschemas, in dem die Beziehungstabelle liegt
      schema: 'public',

      // Ziel-Tabelle, die Freundschaftsanfragen und Beziehungsstatus speichert
      table: 'friend_requests'
```

```
19      },
20      // Callback-Funktion, die bei jeder erkannten Änderung ausgeführt wird
21      () => {
22        // Neuladen der Freundesliste, um neu hinzugefügte oder entfernte
        Freunde
23        // unmittelbar in der Benutzeroberfläche darzustellen
24        fetchFriends(user.id);
25
26        // Aktualisierung aller offenen und eingehenden Freundschaftsanfragen,
27        // sodass angenommene oder abgelehnte Anfragen sofort korrekt
        angezeigt werden
28        fetchRequests(user.id);
29
30        // Synchronisation aller bestehenden Beziehungen, um den globalen
31        // Beziehungszustand (accepted, pending, declined) konsistent zu
        halten
32        fetchAllRelations(user.id);
33      }
34    )
35    // Aktivierung des Realtime-Kanals und Start des kontinuierlichen
        Empfangs
36    // von Datenbankereignissen für die Live-Aktualisierung der
        Benutzeransicht
37    .subscribe();
```

4.2.5.2. Authentifizierung der Benutzer (Supabase Auth, OAuth2)

Die Authentifizierung erfolgt über Supabase Auth als zentrales Identitäts- und Sitzungsmanagement. Supabase übernimmt dabei insbesondere die Verwaltung von Benutzer-Sessions, die Erstellung und Validierung von JSON Web Tokens (JWT), das Handling von Refresh-Tokens sowie sicherheitsrelevante Funktionen wie Passwort-Hashing (bei E-Mail/Passwort-Login) und die Integration externer OAuth-Provider.

4.2.5.2.1. Laden von Benutzer- und Profildaten:

Nach erfolgreicher Authentifizierung wird die eindeutige `user.id` aus der Supabase-Session extrahiert und als primärer Schlüssel für benutzerspezifische Abfragen verwendet. Darauf basierend werden Profildaten (z. B. Benutzername, Avatar) aus einer `profiles`-Tabelle geladen und im lokalen Zustand der Anwendung gehalten, um personalisierte Inhalte und Spielzustände konsistent darstellen zu können.

4.2.5.2.2. Logout:

Der Logout erfolgt über `supabase.auth.signOut()`, wodurch die Session invalidiert und gespeicherte Tokens entfernt werden. Zusätzlich wird der lokale Session-Speicher geleert und der Benutzer in den Login-Bereich der Anwendung zurückgeführt, um einen konsistenten und sicheren Zustand sicherzustellen.

4.2.5.2.3. Sicherheitsaspekte:
Die Übertragung der Tokens erfolgt verschlüsselt über TLS. Google übernimmt im OAuth2-Flow die Identitätsprüfung des Benutzers; Supabase prüft anschließend die Gültigkeit und Signatur der Tokens bei jeder autorisierten Anfrage. Darüber hinaus schützt Row Level Security (RLS) auf Datenbankebene benutzerspezifische Datensätze vor unautorisierten Zugriffen, indem Abfragen und Mutationen nur innerhalb der durch Policies definierten Rechte möglich sind. Im OAuth-basierten Verfahren werden keine Passwörter in der Anwendung gespeichert oder verarbeitet.

4.2.6. Spielmechanik und Logik

Dieser Abschnitt beschreibt die grundlegende Spielmechanik von *WoSamma*. Dabei wird erläutert, wie Spielrunden erstellt werden, wie Mehrspieler-Partien mit Freunden umgesetzt sind und nach welchen Kriterien die Punktevergabe erfolgt.

4.2.6.1. Generierung der Spielrunden

Die Generierung der Spielrunden im Einzelspieler-Modus *„Ganz Österreich“* basiert auf der zufälligen Auswahl einer geografischen Position innerhalb der Landesgrenzen Österreichs. Hierzu wird die Staatsgrenze als Polygon modelliert, das durch eine sequenzielle Liste von Breiten- und Längengraden beschrieben wird:

Quellcode 4.3: Polygonmodell der österreichischen Staatsgrenze zur Positionsgenerierung

```
const austriaPolygon = [
  [47.441885, 13.070504],
  [47.657838, 13.179834],
  [47.755772, 13.020078],
  [47.882686, 13.070962],
  [48.096828, 12.838373],
  ...
  [47.447035, 12.895931],
  [47.442109, 13.077460],
  [47.491624, 13.099476]
];
```

Dieses Polygon definiert den zulässigen Suchraum für die Erzeugung gültiger Zielkoordinaten.

4.2.6.1.1. Punkt-in-Polygon-Prüfung:
Zur Überprüfung, ob eine zufällig generierte Koordinate innerhalb des definierten Polygons liegt, wird der Ray-Casting-Algorithmus eingesetzt. Dabei wird von der zu prüfenden Position aus ein horizontaler Strahl konstruiert und die Anzahl der Schnittpunkte dieses Strahls mit den Polygonkanten bestimmt. Ist die Anzahl der Schnittpunkte ungerade, befindet sich der Punkt innerhalb des Polygons; ist sie gerade, liegt der Punkt außerhalb. Diese Methode gewährleistet eine effiziente und robuste Klassifikation der Koordinaten in Bezug auf die Landesgrenzen.

4.2.6.1.2. Zufällige Koordinatenerzeugung:
Die Generierung einer Spielrunde erfolgt durch wiederholte Erzeugung zufälliger Koordinaten innerhalb eines umschließenden Rechtecks (Bounding Box), das die maximalen und minimalen Breiten- und Längengrade Österreichs abdeckt. Für jede generierte Position wird anschließend mittels der Punkt-in-Polygon-Prüfung verifiziert, ob sich diese innerhalb der tatsächlichen Landesgrenze befindet. Nur gültige Koordinaten werden als Zielposition für eine Spielrunde akzeptiert.

4.2.6.1.3. Abbruchbedingung und Robustheit:
Um eine potenziell unendliche Schleife bei ungünstigen Zufallskonstellationen zu vermeiden, ist eine maximale Anzahl von Iterationen definiert. Wird diese Grenze überschritten, wird die Generierung abgebrochen und der zuletzt berechnete Wert zurückgegeben. Diese Maßnahme stellt die Stabilität der Anwendung sicher und verhindert eine Blockierung des Spielablaufs.

4.2.6.2. Generierung des täglichen Spiels

Das tägliche Spiel wird bei jedem Start der Anwendung initial geprüft und gegebenenfalls neu erzeugt. Hierzu wird zunächst eine Abfrage an das Backend durchgeführt, um festzustellen, ob für das aktuelle Kalenderdatum bereits eine gültige Spielrunde in der Datenbank existiert. Diese Prüfung erfolgt anhand eines datumsbasierten Identifikators, der sicherstellt, dass pro Tag genau eine eindeutige Zielposition verwendet wird.

Ist kein Eintrag für den aktuellen Tag vorhanden, wird die Zielkoordinate nach demselben Verfahren generiert wie im Einzelspieler-Modus. Dabei wird eine zufällige Position innerhalb der definierten Landesgrenzen Österreichs erzeugt und mittels Punkt-in-Polygon-Prüfung validiert. Die validierte Koordinate wird anschließend zusammen mit Metadaten (z. B. Erstellungsdatum, Runden-ID und Spieltyp) persistent in der Datenbank gespeichert.

Existiert bereits ein tägliches Spiel für den aktuellen Tag, wird die gespeicherte Zielposition aus der Datenbank geladen und für alle Spieler identisch verwendet. Durch dieses Vorgehen wird gewährleistet, dass alle Teilnehmer eines Tages unter denselben Bedingungen spielen und die Vergleichbarkeit der Ergebnisse sowie die Integrität der Ranglisten sichergestellt sind.

4.2.6.3. Live-Einladungen für Freundesrunden

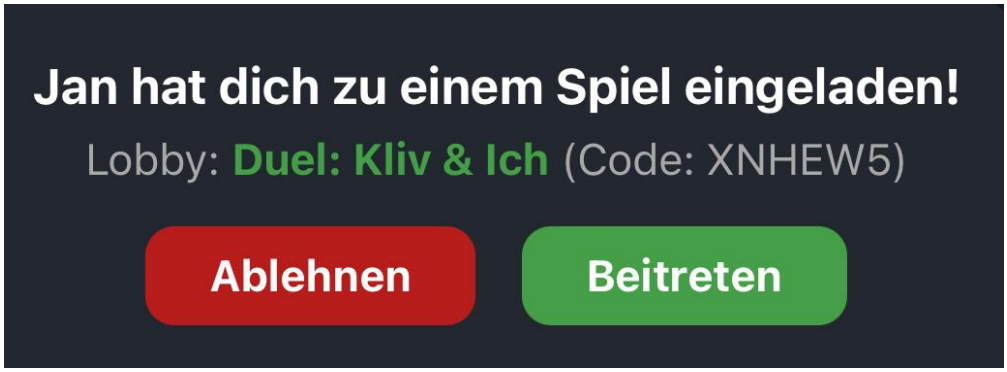

Abbildung 4.13.: Darstellung einer Live-Einladung in der Benutzeroberfläche

Abbildung 4.13 zeigt die visuelle Darstellung einer eingehenden Live-Einladung für den eingeloggten Benutzer. Sobald eine neue Einladung erstellt wird, erscheint automatisch ein Popup mit den relevanten Lobby- und Einladungsinformationen.

Im folgenden Abschnitt wird der zugehörige Code gezeigt, der diese Echtzeit-Funktionalität umsetzt.

Quellcode 4.4: Realtime-Subscription zur Verarbeitung eingehender Lobby-Einladungen

```
useEffect(() => {
  // Vorbedingung:
  // Abbruch der Ausführung, falls keine Benutzer-ID vorhanden ist,
  // da in diesem Fall keine personalisierten Einladungen verarbeitet
    werden können.
  if (!userId) return;

  // Ressourcenverwaltung:
  // Referenz auf das Realtime-Subscription-Objekt zur späteren
    Deregistrierung.
  let subscription: any;

  // Initialisierung:
  // Startet die Echtzeit-Überwachung für neue Lobby-Einladungen.
  async function subscribeInvites() {
    // Kanalaufbau:
    // Erstellung eines Supabase-Realtime-Kanals, der auf neue Einträge
    // in der Tabelle "lobby_invites" für den aktuellen Benutzer reagiert.
    subscription = (supabase.channel('global-lobby-invites-' + userId) as any)
      .on(
        'postgres_changes',
        {
          // Ereignistyp:
          // Reaktion ausschließlich auf das Einfügen neuer Datensätze.
          event: 'insert',

          // Schema:
```

```
26          // Angabe des relevanten Datenbankschemas.
27          schema: 'public',
28
29          // Ziel:
30          // Tabelle, die die Einladungen zu Mehrspieler-Lobbys enthält.
31          table: 'lobby_invites',
32
33          // Filter:
34          // Einschränkung auf Einladungen, die an den aktuell angemeldeten
    Benutzer gerichtet sind.
35          filter: `invited_user_id=eq.${userId}`
36        },
37        // Callback:
38        // Wird ausgeführt, sobald eine neue Einladung empfangen wird.
39        async (payload: any) => {
40          // Datenextraktion:
41          // Auslesen des neu eingefügten Einladungseintrags aus dem
    Ereignisobjekt.
42          const invite = payload.new;
43
44          // Datenabfrage:
45          // Parallelisierte Abfrage der zugehörigen Lobby-Informationen
46          // sowie des Benutzernamens des einladenden Spielers.
47          const [{ data: lobby }, { data: inviter }] = await Promise.all([
48            supabase.from('lobbies')
49              .select('name, code')
50              .eq('id', invite.lobby_id)
51              .single(),
52            supabase.from('profiles')
53              .select('username')
54              .eq('id', invite.invited_by)
55              .single(),
56          ]);
57
58          // UI-Aktualisierung:
59          // Setzt den lokalen Zustand zur Anzeige des Einladungs-Popups
60          // mit allen relevanten Informationen für den Benutzer.
61          setInvitePopup({
62            id: invite.id,
63            lobbyId: invite.lobby_id,
64            lobbyName: lobby?.name || 'Lobby',
65            lobbyCode: lobby?.code || '',
66            inviterName: inviter?.username || 'Ein Freund',
67          });
68
69          // Visualisierung:
70          // Einblendung des Popups mittels Animation zur Verbesserung der
    Benutzererfahrung.
71          Animated.timing(popupAnim, {
72            toValue: 0,
73            duration: 350,
74            useNativeDriver: true
75          }).start();
76        }
```

```
77        )
78        // Aktivierung:
79        // Startet den Realtime-Kanal und beginnt mit dem Empfang von
        Ereignissen.
80        .subscribe();
81    }
82
83    // Initialaufruf:
84    // Start der Echtzeit-Überwachung nach dem Initialisieren der Komponente.
85    subscribeInvites();
86
87    // Aufräumroutine:
88    // Wird beim Verlassen der Komponente ausgeführt, um die
89    // Realtime-Verbindung ordnungsgemäß zu schließen und Ressourcen
        freizugeben.
90    return () => {
91      if (subscription) supabase.removeChannel(subscription);
92    };
93  }, [userId]);
```

4.2.6.4. Punkteberechnung

In diesem Abschnitt wird die Berechnung der Punktzahl im Einzelspieler-Modus von *WoSamma* detailliert beschrieben. Die Punktevergabe basiert auf der Entfernung in Kilometern zwischen der vom Spieler gewählten Position und dem tatsächlichen Standort.

4.2.6.4.1. Grundprinzip:

Die erreichte Punktzahl ist eine monotone, stückweise linear fallende Funktion der Distanz d (in Kilometern) zwischen der vom Spieler gewählten Position und dem tatsächlichen Zielort. Mit abnehmender Distanz erhöht sich die Punktzahl entsprechend. Negative Punktwerte sind grundsätzlich ausgeschlossen; im speziellen Einzelspieler-Modus *„Ganz Österreich“* wird jedoch bei einer Distanz von mehr als 500 km eine negative Punktzahl von bis zu -5000 Punkten vergeben. Diese Maßnahme dient dazu, gezielte Fehlplatzierungen zu verhindern und eine künstliche Erhöhung der Platzierung in der Bestenliste durch zufällige oder absichtlich falsche Eingaben zu unterbinden.

4.2.6.4.2. Punktebereiche und Berechnungsformeln:

- **Perfekter Treffer ($d < 1\,\mathbf{km}$)**
  Maximale Punktzahl von **5000 Punkten**.
  Begründung: Der Spieler befindet sich nahezu exakt auf der korrekten Position.
- **Sehr nah ($1\,\mathbf{km} \leq d < 10\,\mathbf{km}$)**
  Die Punktzahl sinkt linear von 5000 auf 4000 Punkte.
  Formel:

$$P(d) = 5000 - 1000 \cdot \frac{d}{10}$$

Beispiel: Bei $d = 5\,\text{km}$ ergibt sich $P(5) = 4500$ Punkte.

- **Nah ($10\,\text{km} \leq d < 100\,\text{km}$)**
  Die Punktzahl sinkt linear von 4000 auf 1000 Punkte.
  Formel:
  $$P(d) = 4000 - 3000 \cdot \frac{d - 10}{90}$$
  Beispiel: Bei $d = 55\,\text{km}$ ergibt sich $P(55) = 2500$ Punkte.
- **Weit entfernt ($100\,\text{km} \leq d < 500\,\text{km}$)**
  Die Punktzahl sinkt linear von 1000 auf 0 Punkte.
  Formel:
  $$P(d) = 1000 - 1000 \cdot \frac{d - 100}{400}$$
  Beispiel: Bei $d = 300\,\text{km}$ ergibt sich $P(300) = 500$ Punkte.
- **Sehr weit entfernt ($d \geq 500\,\text{km}$)**
  Es werden **0 Punkte** vergeben.
  Begründung: Der Tipp liegt außerhalb des definierten Bewertungsbereichs.

Quellcode 4.5: Distanzberechnung mittels Haversine-Formel

```
// Distanzberechnung unter Berücksichtigung der Erdkrümmung
// Implementierung der Haversine-Formel zur Bestimmung der Groß
    kreisentfernung
// zwischen zwei geografischen Koordinaten (Breiten- und Längengrad).
// Die berechnete Distanz dient als Grundlage für die anschließende
    Punkteberechnung.
function haversineDistance(lat1: number, lon1: number, lat2: number, lon2: number) {

  // Umrechnung von Gradmaß in Bogenmaß, da trigonometrische Funktionen
  // in JavaScript/TypeScript mit Radiantwerten arbeiten
  const toRad = (deg: number) => (deg * Math.PI) / 180;

  // Mittlerer Erdradius in Kilometern gemäß internationalem Referenzwert
  const R = 6371;

  // Berechnung der Differenzen der Breiten- und Längengrade in Radiant
  const dLat = toRad(lat2 - lat1);
  const dLon = toRad(lon2 - lon1);

  // Haversine-Term:
  // Kombination aus Sinus- und Kosinusfunktionen zur Modellierung
  // der sphärischen Geometrie der Erdoberfläche
  const a =
    Math.sin(dLat / 2) * Math.sin(dLat / 2) +
    Math.cos(toRad(lat1)) * Math.cos(toRad(lat2)) *
    Math.sin(dLon / 2) * Math.sin(dLon / 2);

  // Zentrale Winkelentfernung zwischen den beiden Punkten auf der
    Erdoberfläche
  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

  // Berechnung der realen Distanz in Kilometern als Produkt aus
```

```
30    // Erdradius und zentralem Winkel
31    const distance = R * c;
32
33    // Rückgabe der berechneten Distanz, welche anschließend als Eingangsgrö
        ße
34    // für die Punktefunktion verwendet wird
35    return distance;
36  }
```

4.2.7. Benutzerprofil

Dieser Abschnitt beschreibt die Umsetzung des Benutzerprofils innerhalb der Anwendung. Das Benutzerprofil dient zur Darstellung spielrelevanter Informationen sowie zur Verwaltung von freigeschalteten Inhalten und individuellen Einstellungen.

4.2.7.1. Profilanpassung

Beim Öffnen des Benutzerprofils werden die gespeicherten Basisdaten des angemeldeten Nutzers aus der Datenbank geladen und im lokalen Zustand der Anwendung initialisiert. Hierzu zählen insbesondere der Benutzername, die zugewiesene Rolle sowie individuelle Profileinstellungen. Zusätzlich wird ermittelt, welches Profilbild aktuell als aktiv markiert ist und welche Profilbilder dem Benutzer grundsätzlich zur Verfügung stehen beziehungsweise bereits freigeschaltet wurden.

Die Auswahl eines neuen Profilbilds erfolgt über einen konsistenten Aktualisierungsmechanismus. Zunächst werden alle bisher als aktiv markierten Profilbilder des Benutzers zurückgesetzt. Anschließend wird das neu gewählte Profilbild eindeutig als aktiv gesetzt und persistent in der Datenbank gespeichert. Dieses Verfahren stellt sicher, dass zu jedem Zeitpunkt genau ein Profilbild als gültig ausgewählt ist.

Benutzer mit administrativen oder entwicklungsbezogenen Rollen erhalten automatisch Zugriff auf alle verfügbaren Profilbilder. Diese Berechtigung wird serverseitig über rollenbasierte Zugriffsregeln realisiert und clientseitig entsprechend visualisiert.

Werden Profildaten geändert, so erfolgt eine zentrale Speicherung der aktualisierten Attribute in der Datenbank. Hierzu zählen unter anderem der Benutzername, das Geburtsdatum sowie – sofern vom Authentifizierungssystem unterstützt – E-Mail-Adresse und Zugangsdaten. Durch die zentrale Persistenz wird gewährleistet, dass die Änderungen konsistent in allen Ansichten der Anwendung reflektiert werden.

4.2.7.2. Kaufen und Freischalten von Profilinhalten

Der integrierte Shop stellt dem Benutzer alle verfügbaren Profilinhalte, insbesondere Profilbilder und Zubehör, in einer strukturierten Übersicht dar. Bereits freigeschaltete Inhalte werden visuell von noch nicht erworbenen Elementen unterschieden, um den aktuellen Besitzstatus transparent darzustellen.

Der Erwerb eines Profilinhalts erfolgt über einen simulierten Kaufprozess ohne Anbindung an ein externes Bezahlsystem. Bei Bestätigung des Kaufs wird das ausgewählte Element dem Benutzer dauerhaft in der Datenbank zugeordnet. Diese Zuordnung wird in einer Besitzrelation gespeichert, die den Benutzer eindeutig mit den freigeschalteten Inhalten verknüpft.

Nach erfolgreichem Erwerb werden die freigeschalteten Inhalte unmittelbar im Profilbereich verfügbar gemacht und können dort als aktive Darstellungselemente ausgewählt werden. Ein vergleichbarer Mechanismus wird für Zubehör verwendet, wobei der Besitzstatus ebenfalls persistent gespeichert und in der Benutzeroberfläche entsprechend visualisiert wird.

Im öffentlichen Profil eines Benutzers wird stets das aktuell aktivierte Profilbild angezeigt. Diese Darstellung basiert auf der zuletzt im privaten Profil vorgenommenen Auswahl und gewährleistet eine konsistente Repräsentation des Benutzers innerhalb der Anwendung.

4.2.8. Freundesystem

Das Freundesystem ermöglicht die soziale Interaktion zwischen den Benutzern der Anwendung. Es dient dazu, Spieler miteinander zu vernetzen, gemeinsame Spielrunden zu starten und den Austausch innerhalb der Anwendung zu fördern.

4.2.8.1. Freundschaftsanfragen

Die Anwendung stellt eine Suchfunktion zur Verfügung, über die Benutzer gezielt nach anderen Spielern suchen und Freundschaftsanfragen initiieren können. Beim Senden einer Anfrage wird in der Datenbank eine neue Beziehung zwischen den beiden Benutzern angelegt und initial mit dem Status `pending` versehen.

Vor dem Erstellen einer Anfrage erfolgt eine serverseitige Validierung, um sicherzustellen, dass zwischen den beiden Benutzern noch keine bestehende Beziehung vorliegt und dass der Empfänger den Erhalt von Freundschaftsanfragen nicht deaktiviert hat. Diese Prüfungen verhindern redundante Einträge und unerwünschte Kontaktaufnahmen.

Eingehende Anfragen werden dem Benutzer in einer konsolidierten Übersicht angezeigt. Der Benutzer kann jede Anfrage entweder annehmen oder ablehnen, wodurch der Status der Beziehung entsprechend auf `accepted` beziehungsweise `declined` gesetzt wird. Die Listen der bestehenden Freundschaften und offenen Anfragen werden über eine Echtzeit-Synchronisation automatisch aktualisiert, sodass Statusänderungen unmittelbar in der Benutzeroberfläche reflektiert werden.

4.2.8.2. Herausfordern von Freunden

Ausgehend von der Freundesliste kann ein Benutzer einen direkten Mehrspieler-Vergleich initiieren. Hierzu wird eine private Lobby erzeugt, in der der initiierende Benutzer als Host eingetragen wird. Der ausgewählte Freund wird über einen Einladungsmechanismus mit dieser Lobby verknüpft.

Zusätzlich wird eine Beitrittsoption bereitgestellt, sofern sich der Freund in einer offenen, öffentlichen Lobby befindet und dort freie Teilnehmerplätze verfügbar sind. Einladungen werden als eigenständige Datensätze in der Datenbank geführt und können vom Empfänger explizit angenommen oder abgelehnt werden. Dieses Verfahren stellt sicher, dass der Beitritt zu einer Spielrunde stets kontrolliert und nachvollziehbar erfolgt.

4.2.8.3. Chatfunktion

Die Chatfunktion ist als bidirektionale 1-zu-1-Kommunikation zwischen zwei befreundeten Benutzern realisiert. Nachrichten werden in einer dedizierten Nachrichtentabelle persistent gespeichert und über Echtzeit-Mechanismen an beide Teilnehmer synchronisiert.

Beim Öffnen eines Chatfensters werden alle vorhandenen Nachrichten beider Benutzer aus der Datenbank geladen. Neu eingehende Nachrichten werden unmittelbar in der Benutzeroberfläche angezeigt, wodurch eine kontinuierliche und nahezu verzögerungsfreie Kommunikation ermöglicht wird.

4.2.8.4. Anzeigen von Benutzerprofilen

Benutzerprofile können sowohl aus der Freundesliste als auch aus spielbezogenen Kontexten, wie beispielsweise Lobbys oder Bestenlisten, aufgerufen werden. Das öffentliche Profil stellt grundlegende Informationen wie den Benutzernamen, die zugewiesene Rolle sowie ausgewählte statistische Kennzahlen dar.

Darüber hinaus wird das aktuell aktive Profilbild des Benutzers angezeigt. Diese Darstellung basiert auf der zuletzt im privaten Profil vorgenommenen Auswahl und gewährleistet eine konsistente Repräsentation des Benutzers über alle Ansichten der Anwendung hinweg.

4.2.9. Integration der Google APIs

In diesem Abschnitt wird die technische Einbindung der externen Google-Dienste beschrieben, die zur Darstellung, Interaktion und Auswertung geografischer Inhalte innerhalb der Anwendung verwendet werden. Die Google Maps API und die Google Street View API bilden die visuelle und funktionale Grundlage für die Spielmechanik, indem sie sowohl die kartografische Interaktion als auch die realitätsnahe Darstellung der Spielorte ermöglichen. Der Fokus liegt auf der Architektur der Einbindung, der Kommunikation zwischen WebView und nativer Anwendung sowie auf der Validierung und Synchronisation der bereitgestellten Geodaten.

4.2.9.1. Verwendung der Google Maps API

Die Anwendung integriert die Google Maps API zur Darstellung und Interaktion mit geografischen Karteninhalten. Der zugehörige API-Schlüssel wird zur Laufzeit aus einer zentralen Konfigurationstabelle im Backend geladen und im Arbeitsspeicher der Anwendung zwischengespeichert. Durch dieses Caching wird vermieden, dass der Schlüssel bei jedem Wechsel

der Benutzeransicht erneut aus der Datenbank abgerufen werden muss, wodurch sowohl die Netzwerklast als auch die Latenz reduziert werden.

Die Karten werden nicht als native Kartenkomponenten gerendert, sondern als Web-Inhalte in einer *WebView*. Hierzu wird dynamisch eine HTML-Seite erzeugt, in die das Google Maps JavaScript SDK eingebunden wird. Diese Seite wird innerhalb der mobilen Anwendung ausgeführt und ermöglicht eine plattformunabhängige Nutzung identischer Kartenlogik auf iOS und Android.

Für die Umsetzung der Spielmechanik werden unterschiedliche Kartenvarianten erzeugt, die jeweils auf spezifische Interaktionsszenarien ausgelegt sind. Hierzu zählen insbesondere sogenannte *Guess-Karten*, welche das Setzen eines Markers zur Abgabe eines Spieler-Tipps ermöglichen, sowie *Ergebnis-Karten*, die die gewählte Position und den tatsächlichen Zielstandort simultan visualisieren.

Benutzerinteraktionen innerhalb der WebView, wie das Platzieren eines Markers, das Verändern des Zoom- und Kartenfokus oder das Auslösen von Ereignissen, werden über einen Nachrichtenkanal an die native Anwendung zurückgemeldet. Diese bidirektionale Kommunikation stellt sicher, dass die Benutzeroberfläche und die zugrunde liegende Spiel- und Auswertungslogik stets synchron zum aktuellen Kartenzustand bleiben.

4.2.9.2. Einbindung der Street-View-Bilder

Die Einbindung der Street-View-Funktionalität erfolgt ebenfalls über das Google Maps JavaScript SDK innerhalb einer WebView. Hierbei wird ein Street-View-Panorama erzeugt, das dem Benutzer eine 360-Grad-Ansicht des jeweiligen Spielorts bereitstellt und die visuelle Grundlage für das spielerische Erkundungskonzept bildet.

Zur Reduktion visueller Komplexität werden zentrale Benutzeroberflächenelemente des Street-View-Panoramas, wie beispielsweise Navigations- und Zoom-Steuerelemente, weitgehend ausgeblendet. Dadurch wird der Fokus des Benutzers auf die Umgebung und die räumliche Orientierung innerhalb der Spielansicht gelenkt.

Vor der Initialisierung eines Panoramas wird geprüft, ob für die gewünschte Region ein gültiger Street-View-Datenpunkt verfügbar ist. Diese Prüfung erfolgt über eine Kombination aus Street-View-Suchanfragen und, falls erforderlich, zusätzlichen Validierungsmechanismen wie regionsbasierten oder polygonalen Verfügbarkeitsprüfungen. Wird kein geeigneter Punkt gefunden, wird entweder eine alternative Position gesucht oder dem Benutzer eine entsprechende Fehlermeldung angezeigt.

Die aktuelle Kameraausrichtung des Panoramas (Heading) wird in regelmäßigen Abständen aus der WebView an die native Anwendung übertragen. Diese Information wird verwendet, um zusätzliche Overlays, wie beispielsweise einen Kompass, synchron zur Blickrichtung des Benutzers darzustellen und somit eine konsistente und kohärente Benutzererfahrung zu gewährleisten.

5. Zusammenfassung und Ausblick

5.1. Zusammenfassung

Zusammenfassend war diese Diplomarbeit ein sehr lehrreiches Projekt, bei dem wir viele neue Erfahrungen gemacht haben. ...

5.2. Ausblick

I. Literaturverzeichnis

[1] Eckert, Michael: *Apple iOS*, April 2025. Online im Internet: URL: `https://www.computerweekly.com/de/definition/Apple-iOS`.

[2] Deinhard, Florian: *Was ist iOS?*, Juni 2025. Online im Internet: URL: `https://www.it-schulungen.com/wir-ueber-uns/wissensblog/was-ist-ios.html`.

[3] Redaktion, IONOS: *Der Weg zur eigenen App - Teil 5.2: Eine iOS-App veröffentlichen*, Juli 2021. Online im Internet: URL: `https://www.ionos.at/digitalguide/websites/web-entwicklung/die-eigene-app-entwickeln-eine-ios-app-veroeffentlichen/`.

[4] Flemming, Constatin: *Was ist Android? Das Betriebssystem von Google im Check*, März 2025. Online im Internet: URL: `https://www.vodafone.de/featured/smartphones-tablets/was-ist-android-das-betriebssystem-von-google-im-check/#/`.

[5] Redaktion, IONOS: *Eine eigene Native Mobile App entwickeln – Teil 5.1: Im Google Play Store eine App hochladen*, Februar 2020. Online im Internet: URL: `https://www.ionos.at/digitalguide/websites/web-entwicklung/die-eigene-app-entwickeln-android-app-veroeffentlichen/`.

[6] team denvo: *Native vs. Cross-Plattform App: Welcher Weg ist der Richtige für Ihre Idee?*, November 2023. Online im Internet: URL: `https://www.denovo.at/blog/native-vs-cross-plattform-app-welcher-weg-ist-der-richtige-fuer-ihre-idee`.

[7] Hahn, Lennart: *Cross-Platform App – Plattformübergreifende Entwicklung mit Flutter, React Native & Co.*, Januar 2026. Online im Internet: URL: `https://www.itportal24.de/ratgeber/cross-platform-app`.

[8] Große, Vera: *Was ist eine Cross-Platform App?*, März 2025. Online im Internet: URL: `https://www.knguru.de/blog/was-ist-eine-cross-platform-app`.

[9] *Dark vs. Light Mode: Ästhetik, Nutzbarkeit & Energieeffizienz erklärt.* Online im Internet: URL: `https://www.publizer.de/newsroom/dark-mode-vs-light-mode-aesthetik-funktionalitaet-und-energieeffizienz-2893524`.

[10] Gusev, Alexander: *Top Mobile App Design Trends You Should Watch For In 2025*, Oktober 2025. Online im Internet: URL: `https://natively.dev/blog/top-mobile-app-design-trends-2025`.

[11] Redaktion, IONOS: *Was bedeutet Responsive Design?*, Oktober 2022. Online im Internet: URL: `https://www.ionos.at/digitalguide/websites/webdesign/was-bedeutet-responsive-design/`.

[12] Holcombe, Jeremy: *Responsive vs. Adaptive: So wählst du den richtigen Design-Ansatz*, August 2023. Online im Internet: URL: `https://kinsta.com/de/blog/responsive-vs-adaptiv/`.

[13] Cooper, Emily: *Designing mobile apps with accessibility in mind*, May 2024. Online im Internet: URL: `https://www.uxmatters.com/mt/archives/2024/05/designing-mobile-apps-with-accessibility-in-mind.php`.

[14] *Accessibility*. Online im Internet: URL: https://m2.material.io/design/usability/accessibility.html#understanding-accessibility.

[15] Heine, Laura: *Informationsarchitektur in UX - Best Practices*, November 2021. Online im Internet: URL: https://www.b13.com/de/blog/informationsarchitektur-in-ux-best-practices.

[16] Bauer, Anna: *Informationsarchitektur verstehen - oder war es doch die Sitemap?*, November 2025. Online im Internet: URL: https://www.eresult.de/blog/ux-design/blog-informationsarchitektur-ux/.

[17] *Wireframe vs. Prototype - Was ist der Unterschied?*, Juni 2017. Online im Internet: URL: https://www.popwebdesign.de/popart_blog/de/2017/06/wireframe-vs-prototype-was-ist-der-unterschied/.

[18] *Was ist der Unterschied zwischen Wireframes, Prototypen und Mockups?*, Juni 2024. Online im Internet: URL: https://www.justinmind.com/de/wireframe/unterschied-wireframe-vs-prototyp-vs-mockup.

[19] *Wireframe vs. Prototype*. Online im Internet: URL: https://miro.com/de/wireframing/wireframe-vs-prototyp/#high-fidelity-wireframe-vs.-prototyp.

[20] *Low Fidelity vs. High Fidelity Wireframes: Was ist der Unterschied?*, Juni 2024. Online im Internet: URL: https://www.justinmind.com/de/wireframe/low-fidelity-vs-high-fidelity-wireframing-ist-papier-tot.

[21] Olav: *Visuelles Design - Was ist Visuelles Design?*, 2025. Online im Internet: URL: https://toolmaster.ch/visuelles-design-was-ist-visuelles-design/.

[22] Hulatt, Lily: *Visual Design*, November 2024. Online im Internet: URL: https://www.studysmarter.de/schule/kunst/grafikdesign-kunst/visual-design/.

[23] Gupta, Ankit: *Markt für die Entwicklung mobiler Apps*, Oktober 2025. Online im Internet: URL: https://www.marketresearchfuture.com/de/reports/mobile-app-development-market-1752.

[24] Boadum, Leslie: *So funktioniert erfolgreiche In-App-Werbung*, Januar 2023. Online im Internet: URL: https://blog.hubspot.de/marketing/in-app-werbung.

[25] Große, Vera: *In-App-Werbung: Grundlagen, Vorteile und Formate*, Oktober 2025. Online im Internet: URL: https://www.knguru.de/blog/in-app-werbung.

[26] Homsi, Abdullah: *Mobile ad networks: a complete guide to app monetization in 2025*, October 2025. Online im Internet: URL: https://yango-ads.com/blog/mobile-ad-networks-a-complete-guide-to-app-monetization-in-2025-yango-ads.

[27] Leitherer, Johanna: *Sponsoring als Marketinginstrument einsetzen*, März 2018. Online im Internet: URL: https://www.springerprofessional.de/sponsoring/marketingkommunikation/sponsoring-als-marketinginstrument/15516210?

[28] Experts, Infatica SDK: *10 app monetization models explained (with pros & cons)*, June 2025. Online im Internet: URL: https://www.springerprofessional.de/sponsoring/marketingkommunikation/sponsoring-als-marketinginstrument/15516210?

[29] Zahid, Reeba: *Mobile app monetization strategies for high revenue*, February 2025. Online im Internet: URL: https://tanbits.com/blog/mobile-app-monetization-strategies-for-high-revenue/.

[30] Warcholinski, Matt: *Was ist React Native? Alles was Sie für 2025 wissen müssen*, Mai 2025. Online im Internet: URL: https://brainhub.eu/de/library/was-ist-react-native.

[31] Talend, Inc.: *Was ist eine API? – API (Application Programming Interface) einfach erklärt*, 2025. Online im Internet: URL: https://www.talend.com/de/resources/was-ist-eine-api/.

[32] Kosinski, Matthew and IBM Think: *What is a database?*, 2025. Online im Internet: URL: https://www.ibm.com/think/topics/database.

[33] Corporation, Oracle: *Was ist eine relationale Datenbank? (RDBMS)?*, Juni 2021. Online im Internet: URL: https://www.oracle.com/de/database/what-is-a-relational-database/.

[34] Software, DB Engines / Redgate: *Normalisierung*, 2025. Online im Internet: URL: https://db-engines.com/de/article/Normalisierung.

[35] Medienpalast: *Real Time Data – Echtzeitdaten*, 2025. Online im Internet: URL: https://www.medienpalast.net/glossar/begriff/real-time-data/.

[36] Technologies, Akamai: *Horizontale und vertikale Skalierung im Vergleich*, 2026. Online im Internet: URL: https://www.akamai.com/de/glossary/what-is-horizontal-scaling-vs-vertical-scaling.

[37] Ugurcu, Mario: *DevOps-Tool Teil 1: Die Bedeutung von Lastverteilung, CDNs und automatischem Skalieren in IaaS*, März 2024. Online im Internet: URL: https://teamtakt.de/devops-tool-teil-1-die-bedeutung-von-lastverteilung-cdns-und-automatischem-skalie

[38] Redaktion, StudySmarter: *Latenz und Durchsatz – Definition & Beispiele*, September 2024. Online im Internet: URL: https://www.studysmarter.de/schule/informatik/technische-informatik/latenz-und-durchsatz/.

[39] Chauhan, Anshul: *Overview of supabase backend as a service platform*, July 2024. Online im Internet: URL: https://medium.com/@anshuldevx/overview-of-supabase-backend-as-a-service-platform-e192da9a369c.

[40] Deinhard, Florian: *Welche modernen Authentifizierungsmöglichkeiten gibt es?*, Juni 2025. Online im Internet: URL: https://www.it-schulungen.com/wir-ueber-uns/wissensblog/welche-modernen-authentifizierungsmoeglichkeiten-gibt-es.html.

[41] Kaspersky: *Was ist Datenverschlüsselung?*, 2025. Online im Internet: URL: https://www.kaspersky.de/resource-center/definitions/encryption.

[42] Proßnegg, Sabine: *Mobile Apps und Informationspflichten*, Oktober 2017. Online im Internet: URL: https://www.onlinesicherheit.gv.at/Services/Technologie-Schwerpunkte/Mobile-Apps/Mobile-Apps-und-Informationspflichten.html.

[43] Gruppe, HWS: *On Premise, in die Cloud oder doch hybrid?*, 2025. Online im Internet: URL: `https://hws-gruppe.de/on-premises-in-die-cloud-oder-doch-hybrid/`.

[44] Susnjara, Stephanie and Ian Smalley: *What is hybrid cloud architecture?*, 2025. Online im Internet: URL: `https://www.ibm.com/think/topics/hybrid-cloud-architecture`.

[45] Corporation, Microsoft: *Was ist Infrastructure as a Service (IaaS)?*, 2025. Online im Internet: URL: `https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-iaas`.

[46] Corporation, Microsoft: *Was ist Platform-as-a-Service (PaaS)?*, 2025. Online im Internet: URL: `https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-paas`.

[47] Corporation, IBM: *Was ist Function as a Service (FaaS)?*, 2025. Online im Internet: URL: `https://www.ibm.com/de-de/think/topics/faas`.

[48] Red Hat, Inc.: *What is serverless computing?*, 2020. Online im Internet: URL: `https://www.redhat.com/de/topics/cloud-native-apps/what-is-serverless`.

[49] Research, Appinio: *Zielgruppenanalyse: In 7 Schritten zum Erfolg*, März 2024. Online im Internet: URL: `https://www.appinio.com/de/blog/marktforschung/zielgruppenanalyse`.

[50] *Zielgruppenanalyse so definierst du deine Besucher und Kunden*, Mai 2025. Online im Internet: URL: `https://www.webdesign-journal.de/zielgruppenanalyse/`.

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Quellcodeverzeichnis

V. Akronyme

APIs Application Programming Interfaces. 29

VI. Glossar

API Application Programming Interface. Eine Schnittstelle, über die verschiedene Softwarekomponenten oder externe Dienste miteinander kommunizieren und Funktionen austauschen können.. 41, 42

CSS Cascading Style Sheets. Stylesheet-Sprache zur Gestaltung von Layout, Farben, Abständen und visuellen Eigenschaften von Benutzeroberflächen in Webanwendungen.. 62

Entity-Relationship-Diagramm (ERD) Grafische Darstellung eines Datenbankmodells, die Entitäten, deren Attribute sowie die Beziehungen zwischen ihnen zeigt. Es dient der Planung und Strukturierung relationaler Datenbanken.. 64

ES6-Features Erweiterungen von ECMAScript 6 (ES6), die moderne JavaScript-Funktionen wie Klassen, Module, Arrow Functions und verbesserte Variablendeklarationen bereitstellen.. 29

GPS Global Positioning System, ein satellitengestütztes Navigationssystem zur Bestimmung von Position, Geschwindigkeit und Zeit auf der Erde.. 15

Overlay Grafisches Element, das über anderen Inhalten angezeigt wird, ohne deren Struktur zu verändern, z. B. Marker, Labels oder UI-Komponenten auf einer Karte.. 53, 61, 62

Publish-Subscribe-Modell Ein Kommunikationsmuster, bei dem Nachrichten von einem Sender (Publisher) an einen zentralen Kanal gesendet werden, ohne die Empfänger direkt zu kennen. Interessierte Empfänger (Subscriber) abonnieren diesen Kanal und erhalten relevante Ereignisse automatisch. Dieses Modell ermöglicht eine lose Kopplung, skalierbare Echtzeitkommunikation und effiziente Synchronisation zwischen mehreren Systemkomponenten. 66

UI User Interface (Benutzeroberfläche). Bezeichnet die sichtbaren und interaktiven Elemente einer Anwendung, über die Nutzer mit dem System interagieren.. 47, 48

A. Anhang

A.1. Arbeitsteilung

Kurze Beschreibung, wer was gemacht hat (Überblick).

A.2. Kapitelverzeichnis

Kapitel	Editor
2.2 Spezifische Ausgangslage	Max Mustermann
?? ??	Mex Musterjuan

Tabelle A.1.: Kapitelverzeichnis

A.3. Projekttagebücher

A.3.1. Projekttagebuch Max Mustermann

Tag	Zeit	kumulativ	Fortschritt
Mo 28.11.16	2h	2h	Besprechnung der Programmanforderungen
Di 29.11.16	3h	5h	Datenbankmodell erstellt
Mi 30.11.16	1h	6h	Datenbankmodellüberarbeitet
Do 01.12.16	3h	9h	Pflichtenheft erstellt

Tabelle A.2.: Arbeitstagebuch Mustermann

A.3.2. Projekttagebuch Laurenz Pichler

Datum	Zeit	kumulativ	Fortschritt
04.06.2025	18:00–20:45	2,75h	React-Native und WebSocket Technologie gelernt
16.06.2025	19:00–21:00	4,75h	PostgreSQL Einarbeitung und Vergleich zu MySQL
26.06.2025	08:30–12:30	8,75h	Projektplanung
27.06.2025	20:00–21:00	9,75h	Jira Ticket Backlog erstellt
01.07.2025	08:00–09:15	11h	Pflichtenheft bearbeitet
01.07.2025	09:15–10:00	11,75h	Projektstruktur aufgesetzt
01.07.2025	10:00–10:30	12,25h	Teammeeting
01.07.2025	10:30–11:30	13,25h	Login/Register Frontend erstellt
01.07.2025	11:30–13:30	15,25h	Supabase eingerichtet

01.07.2025	14:00–17:30	18,75h	Register/Login entwickelt
01.07.2025	18:00–20:15	21h	Register/Login finalisiert
02.07.2025	07:00–12:00	26h	Freunde-System implementiert
02.07.2025	12:30–16:30	30h	Freunde-System erweitert
03.07.2025	07:00–12:00	35h	Freunde-System optimiert
03.07.2025	12:30–16:30	39h	Freunde-System abgeschlossen
07.07.2025	07:00–12:00	44h	Neuigkeiten-System erstellt
07.07.2025	12:30–16:30	48h	Neuigkeiten erweitert
08.07.2025	07:00–12:00	53h	Lobby-System entwickelt
08.07.2025	12:30–16:30	57h	Lobby erweitert
09.07.2025	07:00–12:00	62h	Realtime Lobby-Einladungen
09.07.2025	12:30–16:30	66h	Spieler aus Lobby entfernen
10.07.2025	07:00–12:00	71h	Google API integriert
10.07.2025	12:30–16:30	75h	Einzelspieler erweitert
11.07.2025	18:00–19:30	76,5h	Supabase Bucket erstellt
14.07.2025	07:00–12:00	81,5h	Policies + Profilbilder implementiert
14.07.2025	12:30–16:30	85,5h	Einzelspieler verbessert
14.07.2025	17:30–20:00	88h	Österreich-Polygon erstellt
15.07.2025	07:00–10:00	91h	Freunde-System überarbeitet
15.07.2025	10:00–12:30	93,5h	Teammeeting
15.07.2025	12:30–16:30	97,5h	Rollen/Berechtigungen hinzugefügt
16.07.2025	07:00–12:00	102,5h	Admin/Developer Rechte erweitert
16.07.2025	12:30–16:30	106,5h	Ticket-System Freunde melden
17.07.2025	07:00–12:00	111,5h	Admin Report-System
17.07.2025	12:30–16:30	115,5h	Account Löschung automatisiert
21.07.2025	07:00–12:00	120,5h	Scoreboard erstellt
21.07.2025	12:30–16:30	124,5h	Daily Level programmiert
22.07.2025	07:00–12:00	129,5h	Daily Level erweitert
22.07.2025	12:30–16:30	133,5h	Daily Scoreboard
23.07.2025	07:00–12:00	138,5h	Timer + Bugfixes
23.07.2025	12:30–16:30	142,5h	Multiplayer Lobby verbessert
24.07.2025	07:00–12:00	147,5h	Shop begonnen
24.07.2025	12:30–16:30	151,5h	Shop fertiggestellt
25.07.2025	10:00–15:30	157h	APK erstellt + Debugging
28.07.2025	07:00–12:00	162h	Multiplayer Synchronisation
28.07.2025	12:30–16:30	166h	Gleiche Location Multiplayer
29.07.2025	07:00–12:00	171h	Realtime Timer Logik
29.07.2025	12:30–16:30	175h	Multiplayer Result Screen
07.08.2025	08:00–13:00	180h	Bugfixes + Google Login + Tests
09.09.2025	14:55–15:30	180,58h	Kick-Off Meeting
18.09.2025	10:45–11:15	181,08h	Pflichtenheft Meeting
09.10.2025	10:45–11:30	181,83h	UI Kontroll Meeting
06.11.2025	13:55–15:45	183,67h	LaTeX Setup
20.11.2025	08:00–09:00	184,67h	Backend Kontrolle
25.11.2025	08:00–09:00	185,67h	Ausgangslage formuliert

25.11.2025	20:00–21:30	187,17h	Recherche Grundlagen
29.11.2025	17:00–19:30	189,67h	Weitere Recherche
24.12.2025	08:00–11:15	192,92h	Theorie geschrieben begonnen
02.01.2026	18:00–21:00	195,92h	Theorie fertiggestellt
16.01.2026	14:00–17:00	198,92h	Literatur und Fehler verbessert
18.01.2026	08:00–09:30	200,42h	Theorie überarbeitet
27.01.2026	12:30–13:30	201,42h	Diplomarbeits Meeting
28.01.2026	16:30–22:45	207,67h	Technische Umsetzung dokumentiert

Tabelle A.3.: Projekttagebuch Laurenz Pichler

A.4. Besprechungsprotokolle

. . . Hier können auch pdf Dateien eingebunden werden!

htl bildung mit zukunft	**HTL Krems** **Höhere Lehranstalt für Informationstechnologie** **Ausbildungsschwerpunkt**	**Reife- und Diplomprüfung**

Betreuungsprotokoll zur Diplomarbeit lfd. Nr.:

Themenstellung:
Kandidaten/Kandidatinnen:

Jahrgang:
Betreuer/in:
Ort:
Datum:
Zeit:

Besprechungsinhalt:

Name	Notiz

Aufgaben:

Name	Notiz	zu erledigen bis

A.5. Besprechungsprotokolle 1

htl bildung mit zukunft	**HTL Krems** **Höhere Lehranstalt für Informationstechnologie** **Ausbildungsschwerpunkt**	**Reife- und Diplomprüfung**

Betreuungsprotokoll zur Diplomarbeit **lfd. Nr.:**

Themenstellung:
Kandidaten/Kandidatinnen:

Jahrgang:
Betreuer/in:
Ort:
Datum:
Zeit:

Besprechungsinhalt:

Name	Notiz

Aufgaben:

Name	Notiz	zu erledigen bis

A.6. Datenträgerbeschreibung