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 den Theoretischen Grundlagen werden die zentralen Grundlagen für die Entwicklung der App erläutert.

3.1. App-Design & Monetarisierungskonzepte

Im ersten Teil der Theoretischen Grundlagen werden das Design und die möglichen Monetarisierungskonzepte mobiler Applikationen erläutert.

3.1.1. Plattformwahl & technische Grundlagen

Wenn man sich dazu entscheidet eine mobile Applikation zu entwickeln, muss man sich zuerst die Frage stellen, auf welchen Plattformen die App laufen soll. Die beiden dominierenden Betriebssysteme im mobilen App Bereich sind iOS von Apple und Android von Google. Diese Entscheidung hat nicht nur Auswirkungen auf die technische Umsetzung, sondern auch auf das Design, die Verfügbarkeit und die Monetarisierungsmöglichkeiten der App.

iOS
iOS ist das mobile Betriebssystem von Apple, das die technologische Grundlage für Geräte wie das iPhone und iPad bildet. Es ist speziell für Touch-Bedienung und intuitive Nutzung konzipiert und bekannt für hohe Sicherheitsstandards, eine intuitive Benutzeroberfläche und die nahtlose Integration ins Apple-Ökosystem. Technisch basiert iOS auf einem Unix-ähnlichen System-Kernel namens Darwin, was eine solide, stabile Basis für moderne mobile Anwendungen schafft. Da Apple Hard- und Software eng verzahnt und die Plattform stark kontrolliert, können Updates, Sicherheitsmechanismen und Apple-Dienste über alle unterstützten Geräte sehr einheitlich bereitgestellt werden. Um eine Applikation für iOS Geräte verfügbar zu machen, muss diese über den Apple App Store vertrieben werden, wobei strenge Richtlinien und Prüfprozesse sicherstellen, 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 Linux-basiertes, mobiles Betriebssystem von Google, das hauptsächlich auf Smartphones und Tablets läuft und als Plattform alle System- und Benutzerkomponenten umfasst, also das Linux-Kernel-Betriebssystem, die grafische Oberfläche und die nutzbaren Apps. Android wurde unter der Apache-Open-Source-Lizenz veröffentlicht, was Herstellern und Entwicklern erlaubt, die Software anzupassen oder eigene Varianten zu bauen. Obwohl die Basis offen ist, enthalten die meisten Geräte zusätzliche proprietäre Programme wie Google-Apps, die vorinstalliert sind. Dadurch ist Android heute das am weitesten verbreitete mobile Betriebssystem 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, die sicherstellen sollen, 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 unter anderem in ihrer Systemarchitektur, Designrichtlinien und Gerätevielfalt. Während iOS auf eine begrenzte Anzahl an Endgeräten optimiert ist, existiert im Android-Bereich eine große Vielfalt an Bildschirmgrößen und Hardwarekonfigurationen. Diese Unterschiede erhöhen den Entwicklungs- und Wartungsaufwand bei nativen Anwendungen.

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 Nutzerinnen und 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
Für die Umsetzung des Frontends wurde das Framework React Native eingesetzt. React Native ist ein von Facebook entwickeltes Framework, das es erlaubt, mobile Anwendungen unter Verwendung von JavaScript und React zu erstellen und eine gemeinsame Codebasis für iOS und Android zu nutzen. Dabei werden Benutzeroberflächen aus modularen, wiederverwendbaren Komponenten aufgebaut, die jeweils eigenständige UI-Elemente wie Schaltflächen oder Ansichten repräsentieren und so eine klare Struktur der App gewährleisten. Durch dieses komponentenbasierte System können Entwickelnde plattformspezifische Unterschiede adressieren und zugleich einen einheitlichen Look & Feel über beide Zielplattformen erreichen, da React Native die native Darstellung der Komponenten auf iOS und Android übernimmt. React Native bietet zudem die Möglichkeit, bei Bedarf auf native APIs zuzugreifen, um plattformspezifische Funktionalität zu integrieren. [6] [7] [8]

3.1.2. Trends & Weiterentwicklungen im App-Design

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. Genau aus diesem Grund stellen Dark Mode und Light Mode gemeinsam einen wichtigen Bestandteil einer zeitgemäßen Designstrategie dar. [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]

Wahrnehmbarkeit: 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]

Bedienbarkeit: 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. Grafische Elemente ohne Informationsgehalt oder mit bereits erklärendem Kontext können bewusst ohne zusätzliche Beschreibung eingesetzt werden, um Wiederholungen zu vermeiden.[13] [14]

Testing und Umsetzung
Barrierefreiheit entsteht nicht nur durch „Guidelines lesen“, sondern durch konsequentes Testen. Standard-UI-Komponenten und semantisches Markup sind oft eine stabile Basis, während Custom Widgets schnell mehr Aufwand und Fehlerquellen bringen. In der Praxis sollten zentrale Tasks end-to-end mit aktivierten Accessibility-Funktionen getestet werden, wie Screenreader-Navigation, Fokus-Reihenfolge, große Schrift und Kontrastanpassungen. Zusätzlich sind Tests mit betroffenen Nutzer:innen extrem wertvoll, weil sie reale Nutzungsmuster sichtbar machen, die im Team sonst leicht übersehen werden.

3.1.3. UI/UX-Design & Nutzererlebnis

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

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, jenes ist vor allem bei Websites relevant. [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.

Mockups, Prototypen & Wireframes
Wireframes und Prototypen 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 mit Stakeholdern 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: Da Wireframes kaum Interaktivität und keine realistischen Inhalte enthalten, 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 bis hin zu High-Fidelity-Prototypen, 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]

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]

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.[17][18][19]

Zusammenspiel im Designprozess
Die verschiedenen Design-Artefakte sind keine konkurrierenden Phasen, sondern ergänzen sich. In der Regel beginnt der Designprozess mit Wireframes, die als Fundament dienen. Danach werden Mockups angefertigt um das visuelle Erscheinungsbild zu evaluieren. Sobald Struktur und Funktionalität festgelegt sind, werden diese Entwürfe in Prototypen überführt, um das Nutzungserlebnis realistisch darzustellen und zu testen. In manchen Fällen entstehen auch High-Fidelity-Wireframes, die bereits detaillierter sind, jedoch noch nicht den vollen Funktionsumfang eines Prototyps besitzen. [17][18][19]

Visuelles Design
Visuelles Design bezeichnet die gezielte Gestaltung visueller Elemente mit dem Ziel, digitale Produkte ästhetisch 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 maßgeblich beeinflusst, wie Inhalte wahrgenommen, verstanden und genutzt werden. In der digitalen Welt ist visuelles Design daher ein wesentlicher Bestandteil von Webanwendungen, Software, Marketingplattformen und interaktiven Medien.[20][21]

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.[20][21]

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.[20][21]

Evaluation und Optimierung im Designprozess
Moderne visuelle Designs werden zunehmend daten- und nutzerorientiert entwickelt. Mithilfe von Benutzeranalysen und Usability-Tests lassen sich Designentscheidungen überprüfen und optimieren. Die Wirkung visueller Gestaltung ist somit messbar und kann gezielt zur Verbesserung der Nutzererfahrung und Markenwahrnehmung eingesetzt werden.[20][21]

3.2. Monetarisierung von mobilen Apps

In diesem Kapitel wird der globale App markt und die verschiedene Strategien zur Monetarisierung mobiler Apps vorgestellt.

3.2.1. Grundlagen & Marktüberblick

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.[22]

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. [22]

Nutzerverhalten & Marktstatistiken
Marktanalysen zeigen, dass Nutzer zunehmend bereit sind, für digitale Inhalte und Zusatz-

funktionen 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.[22]

3.2.2. Werbung als Einnahmequelle

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.[23][24]

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.[23][24]

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.[23][24]

Videoanzeigen (inkl. Rewarded Videos)
Videoanzeigen eignen sich besonders gut, um Inhalte emotional und „story-basiert“ zu vermitteln. Man unterscheidet u.,a. zwischen In-Stream- und Outstream-Formaten (je nachdem, ob die Anzeige innerhalb eines Video-Kontexts oder außerhalb davon abgespielt wird). Eine besonders verbreitete Form in Gaming-Apps ist [23][24]

Rewarded Video-Werbung
Nutzer sehen sich freiwillig ein Video an und erhalten dafür eine Belohnung (z.,B. virtuelle Währung, Bonuspunkte oder zusätzliche Inhalte). Dieses Format gilt als nutzerfreundlicher, weil es einen klaren Mehrwert bietet und häufig eine höhere Akzeptanz erreicht als erzwungene Werbeunterbrechungen.[23][24]

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.[23][24]

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.[23][24]

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.[23][24]

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.[25]

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.[25]

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.[25]
- **Smaato**
  Smaato ist eine mobile Werbeplattform mit starkem Fokus auf programmatische Werbung. 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.[25]
- **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.[25]

- **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.[25]

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.[25]

3.2.3. Sponsoring & Partnerschaften

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.[26]

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.[26]

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 Nutzerinnen bzw. Nutzern aufgebaut werden.[26]

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.[26]

3.2.4. Kostenpflichtige App-Modelle

Einmaliger Kauf
Beim Modell des einmaligen Kaufs bezahlen Nutzer vor dem Download der App einen fixen Betrag und erhalten anschließend uneingeschränkten Zugriff auf alle Funktionen. Dieses Monetarisierungsmodell zählt zu den ältesten Ansätzen 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.[27]

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.[27]

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.[27]

3.2.5. In-App-Käufe

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.[27]

Personalisierung
Ein häufiger Anwendungsbereich von In-App-Käufen ist die Personalisierung 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 emotionale Bindung an die App und stellen insbesondere bei engagierten Nutzern eine effektive Einnahmequelle dar.[27]

3.2.6. Spenden & freiwillige Unterstützung

Spenden & Crowdfunding
Ein weiteres Monetarisierungsmodell besteht darin, Nutzer freiwillig um finanzielle Unterstützung zu bitten, ohne ihnen Pflichten oder Käufe aufzuzwingen. Im Unterschied zu klassischen Einnahmemodellen (z. B. Werbung, In-App-Käufe oder Abonnements) basiert dieses Modell auf freiwilligen Beiträgen der Nutzer, die die App oder ihre Entwickler aus Überzeugung unterstützen möchten.

Crowdfunding bietet Entwickler:innen die Möglichkeit, finanzielle Mittel direkt von Nutzer:innen oder Unterstützer:innen zu erhalten, die an die Vision 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. Ergänzend dazu ermöglichen In-App-Spendenfunktionen den Nutzer:innen, die App freiwillig finanziell zu unterstützen. Dieses Finanzierungsmodell eignet sich besonders für Apps mit klar definierten Zielen, sozialen Anliegen oder kreativen Konzepten. Darüber hinaus trägt Crowdfunding nicht nur zur Finanzierung bei, sondern dient auch zur Validierung der App-Idee und zum Aufbau einer frühen, engagierten Nutzerbasis.

3.3. 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.

3.3.1. 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.

3.3.1.1. 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 [28].

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 [28].

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 [28].

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 [28].

3.3.2. 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.

3.3.2.1. APIs

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 [29].

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 [29].

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 [29].

3.3.2.2. 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 [30].

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 [30].

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 [30].

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 [30].

3.3.2.3. 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 [31].

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 [31]:

- **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 [31].

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 [31].

3.3.3. Hosting- und Infrastrukturmodelle

Hosting- und Infrastrukturmodelle im Kontext moderner IT-Systeme beschreiben, wo und wie Rechenressourcen, Speicher und Anwendungen betrieben werden. Grundsätzlich lassen sich dabei drei zentrale Architekturansätze unterscheiden: On-Premise-Infrastrukturen, Cloud-Architekturen und Hybrid-Architekturen [32, 33].

3.3.3.1. On-Premise, Cloud und Hybrid-Architekturen

Hosting- und Infrastrukturmodelle im Kontext moderner IT-Systeme beschreiben, wo und wie Rechenressourcen, Speicher und Anwendungen betrieben werden. Grundsätzlich lassen sich dabei drei zentrale Architekturansätze unterscheiden: On-Premise-Infrastrukturen, Cloud-Architekturen und Hybrid-Architekturen.[32, 33]

Bei einer On-Premise-Architektur werden sämtliche IT-Ressourcen innerhalb des Unternehmens betrieben. Die benötigte Hardware, wie Server, Speichersysteme und Netzwerkkomponenten, befindet sich in eigenen Räumlichkeiten oder Rechenzentren und wird vollständig vom Unternehmen selbst verwaltet. Dieses Modell bietet ein hohes Maß an Kontrolle über Daten, Systeme und Sicherheitsmechanismen, ist jedoch mit erhöhtem finanziellem Aufwand für Anschaffung, Wartung und Betrieb verbunden.

Cloud-Architekturen hingegen basieren auf der Bereitstellung von IT-Ressourcen durch externe Cloud-Anbieter über das Internet. Rechenleistung, Speicher und Softwaredienste werden bedarfsgerecht zur Verfügung gestellt und vom Anbieter verwaltet. Dieses Modell ermöglicht eine hohe Skalierbarkeit und Flexibilität, da Ressourcen dynamisch angepasst werden können. Zudem entfällt für Unternehmen die Notwendigkeit, eigene physische Infrastruktur in vollem Umfang zu betreiben.

Hybrid-Architekturen stellen eine Kombination aus On-Premise- und Cloud-Lösungen dar. In diesem Ansatz werden bestimmte Systeme oder Daten lokal betrieben, während andere Komponenten in einer Cloud-Umgebung angesiedelt sind. Dadurch können Unternehmen sowohl die Kontrolle und Sicherheit lokaler Infrastrukturen als auch die Skalierbarkeit und Effizienz von Cloud-Diensten nutzen. Anwendungen und Daten können dabei je nach Anforderungen zwischen den Umgebungen verteilt oder integriert werden.

Hybrid-Cloud-Architekturen gewinnen insbesondere im Rahmen der digitalen Transformation an Bedeutung, da sie bestehende On-Premise-Systeme mit modernen Cloud-Technologien verbinden. Sie ermöglichen eine flexible und anpassungsfähige IT-Infrastruktur, die sowohl wirtschaftliche als auch technische Vorteile vereint.

3.3.3.2. IaaS, PaaS, FaaS und Serverless

Infrastructure as a Service (IaaS) ist ein Cloud-Computing-Modell, bei dem grundlegende IT-Ressourcen wie Rechenleistung, Speicher und Netzwerk über das Internet bereitgestellt werden. IaaS ermöglicht es Unternehmen, IT-Infrastruktur bedarfsgerecht zu nutzen und zu skalieren, ohne in physische Hardware investieren oder diese verwalten zu müssen. Verbraucher greifen dabei über eine Webschnittstelle oder APIs auf virtualisierte Ressourcen zu und können diese

flexibel konfigurieren, wodurch sich sowohl Kosten als auch Verwaltungsaufwand reduzieren lassen.[34]

Platform as a Service (PaaS) geht über die reine Bereitstellung von Infrastruktur hinaus und stellt zusätzlich eine vollständig verwaltete Entwicklungsumgebung für Anwendungen zur Verfügung. In einem PaaS-Modell übernimmt der Cloud-Provider nicht nur die zugrunde liegende Hardware und Netzwerkinfrastruktur, sondern auch Softwarekomponenten wie Laufzeitumgebungen, Middleware und Entwicklungstools. Dadurch können Entwickler Anwendungen erstellen, testen und bereitstellen, ohne sich um die Komplexität der Infrastruktur und Laufzeitumgebungen kümmern zu müssen.[35]

Function as a Service (FaaS) ist ein spezielles Cloud-Modell, bei dem einzelne Funktionen oder Code-Snippets in Reaktion auf Ereignisse ausgeführt werden, ohne dass Entwickler Server verwalten müssen. Bei FaaS wird der Code durch Ereignisse ausgelöst und skaliert automatisch entsprechend der Nachfrage. Dieses Modell abstrahiert die zugrunde liegende Infrastruktur vollständig und erlaubt es Entwicklern, applikationsspezifische Logik bereitzustellen, ohne sich um Server-Bereitstellung oder -Betrieb kümmern zu müssen.[36]

Serverless bezeichnet ein umfassenderes Cloud-Paradigma, bei dem Serververwaltung, Skalierung und Ressourcenallokation vollständig vom Cloud-Provider übernommen werden. Trotz des Namens laufen Server weiterhin im Hintergrund, sind jedoch für Entwickler unsichtbar. Beim Serverless-Computing konzentrieren sich Entwickler ausschließlich auf die Implementierung von Funktionalität, während der Provider Betriebsaufgaben wie Provisionierung, Skalierung und Wartung übernimmt. Serverless umfasst dabei nicht nur FaaS, sondern auch weitere verwaltete Dienste wie Datenbanken, APIs und ereignisgesteuerte Architekturen, die gemeinsam eine vollständig abstrahierte Laufzeitumgebung bilden.[37]

3.3.3.3. 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.[38]

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.

3.3.3.4. 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.[39]

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.

3.3.3.5. 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.[40]

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.

3.3.3.6. 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 [41].

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.[41]

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.[41]

3.3.3.7. 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.[42]

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.[42]

In Kombination tragen Autoscaling und Lastverteilung dazu bei, Systeme flexibel und robust gegenüber Lastschwankungen zu machen. Während das Autoscaling die Anzahl oder Leis-

tungsfähigkeit 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.[42]

3.3.3.8. 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.[43]

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.[43]

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.[43]

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.[43]

3.3.3.9. 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.[44]

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.[44]

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.[44]

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.[44]

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.[44]

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.[44]

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.[44]

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.[44]

3.3.3.10. 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.[45]

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.[45]

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.[45]

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.[45]

3.3.3.11. 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.[46]

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.[46]

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.[46]

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.

GRAFIKK!!!!!!!!!

4.2.2. Frontend-Implementierung - JAN NOCH UMSCHREIBEN

In diesem Abschnitt wird die Umsetzung der Benutzeroberfläche von *WoSamma* beschrieben. Der Fokus liegt auf der Struktur der Anwendung, der Navigation zwischen verschiedenen Bildschirmen sowie der Interaktion des Benutzers mit der Spielansicht. Besonderes Augenmerk liegt dabei auf der Darstellung von Spielinformationen und der Visualisierung von Spielfortschritten sowohl im Einzelspieler- als auch im Mehrspielermodus.

4.2.2.1. Projektstruktur und Navigation

Die Projektstruktur orientiert sich an bewährten Konzepten von React Native und TypeScript, um eine modulare, wiederverwendbare und wartbare Codebasis zu gewährleisten. Die Navigation zwischen den verschiedenen Bildschirmen, wie Startbildschirm, Spielansicht, Freundeslobby und Profilen, erfolgt über ein Stack- und Tab-basiertes Navigationssystem. Dieses erlaubt einen intuitiven Zugriff auf alle relevanten Funktionen und stellt sicher, dass der Benutzer jederzeit den Überblick über den Spielstatus behält.

4.2.2.2. Spielansicht und Benutzerinteraktion

Die Spielansicht bildet das zentrale Interface für den Spieler und ist in Einzelspieler- und Mehrspielermodus unterteilt.

4.2.2.2.1. Einzelspieler-Modus: Im Einzelspieler-Modus wird dem Benutzer die korrekte Location sowie die vom Spieler geratene Position angezeigt. Die Distanz zwischen der tatsächlichen Location und der gewählten Position wird berechnet und visualisiert, wodurch der Spieler unmittelbar Feedback zu seiner Genauigkeit erhält. Zusätzlich werden die Punkte basierend auf dieser Distanz angezeigt, um den Fortschritt und die Leistung nachvollziehbar zu machen.

4.2.2.2.2. Mehrspieler-Modus: Im Mehrspieler-Modus werden alle Spieler in einer Lobby zusammengeführt. Jeder Spieler setzt seine Pins auf der Karte, welche live an alle Teilnehmer übertragen werden. Zusätzlich wird der richtige Pin angezeigt, um die Vergleichbarkeit der Ergebnisse zu gewährleisten. Ein Vektor zwischen der geratenen Position jedes Spielers und dem tatsächlichen Standort wird dargestellt, und die Distanz sowie die erreichten Punkte werden für alle Spieler übersichtlich angezeigt. Dies ermöglicht ein direktes, kompetitives Spielerlebnis und fördert die Interaktion zwischen den Teilnehmern.

4.2.3. Backend-Anbindung mit Supabase

4.2.3.1. Datenbankstruktur

4.2.3.2. Authentifizierung der Benutzer - Supabase Auth, OAuth2

4.2.4. 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.

4.2.4.1. Generierung der Spielrunden

4.2.4.2. Generierung des täglichen Spiels

4.2.4.3. Live-Einladungen für Freundesrunden

4.2.4.4. Punkteberechnung

4.2.5. 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.

4.2.5.1. Profilanpassung

4.2.5.2. Kaufen und Freischalten von Profilinhalten

4.2.6. 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.

4.2.6.1. Freundschaftsanfragen

4.2.6.2. Herausfordern von Freunden

4.2.6.3. Chatfunktion

4.2.6.4. Anzeigen von Benutzerprofilen

4.2.7. Integration der Google APIs

4.2.7.1. Verwendung der Google Maps API

4.2.7.2. Einbindung der Street-View-Bilder

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

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

[22] 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.

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

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

[25] 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.

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

[27] 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?

[28] 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.

[29] 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/.

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

[31] 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.

[32] 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/.

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

[34] 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.

[35] 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.

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

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

[38] 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/.

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

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

[41] 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.

[42] 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

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

[44] 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.

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

[46] 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.

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

[48] *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

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 Mex Musterjuan

Tag	Zeit	kumulativ	Fortschritt
Mo 28.11.16	2h	2h	Besprechnung der Programmanforderungen

Tabelle A.3.: Arbeitstagebuch Musterjuan

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