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

3.1. App-Design & Monetarisierungskonzepte

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.

`https://www.ionos.at/digitalguide/websites/web-entwicklung/die-eigene-app-entwickeln-`
`https://www.it-schulungen.com/wir-ueber-uns/wissensblog/was-ist-ios.html https://www.`

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.

`https://www.ionos.at/digitalguide/websites/web-entwicklung/die-eigene-app-entwickeln-`
`https://www.vodafone.de/featured/smartphones-tablets/was-ist-android-das-betriebssyste`

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
Bei einer nativen App handelt es sich um eine Anwendung, die speziell für das Betriebssystem eines mobilen Endgeräts wie iOS oder Android konzipiert und entwickelt wurde. Native Apps werden über die an das jeweilige System gekoppelten App Stores bereitgestellt und können direkt auf die Hardware und Systemfunktionen des Geräts zugreifen, um Ressourcen wie Arbeitsspeicher, Kamera, GPS oder andere Sensoren optimal zu nutzen. Durch diese enge Integration mit dem Betriebssystem bieten native Apps eine gute Performance und hohe Usability. Zudem können sie, abhängig von den Funktionen des Geräts, auch systeminterne Möglichkeiten wie Push-Benachrichtigungen ansteuern. Allerdings erfordert die plattformspezifische Entwicklung separate Versionen für unterschiedliche Betriebssysteme, was den Entwicklungsaufwand erhöht. `https://de.ryte.com/wiki/Native_App/`

Cross-Platform
Bei der Cross-Platform-App-Entwicklung wird eine einzige Codebasis genutzt, die mittels spezieller Frameworks wie Flutter, React Native oder Xamarin in die jeweilige native Sprache für verschiedene Betriebssysteme wie iOS und Android übersetzt wird. Dadurch lässt sich dieselbe Anwendung auf mehreren Plattformen bereitstellen, ohne den Code für jede Zielumgebung separat schreiben zu müssen. Die gemeinsame Codebasis führt zu einem geringeren Entwicklungsaufwand und ermöglicht eine schnellere Markteinführung sowie eine einfachere Wartung, da Änderungen am Code nur einmal vorgenommen werden müssen, um sie auf allen unterstützten Systemen verfügbar zu machen. Zudem kann eine Cross-Platform-App durch diesen Ansatz viele native Funktionen nutzen und sich in vielerlei Hinsicht wie eine native App anfühlen, auch wenn sie nicht für jede Plattform vollständig eigenständig entwickelt wurde.

`https://www.itportal24.de/ratgeber/cross-platform-app` `https://www.knguru.de/blog/was-ist-eine-cross-platform-app`

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.

`https://www.knguru.de/blog/app-entwicklung-mit-react-native-vor-und-nachteile`
`https://reactnative.dev/`

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. `https://techwerk.io/blogs/ux-trends-2025` `https://www.knguru.de/blog/ux-design-trends-fur-erfolgreiche-digitale-produkte` `https://www.publizer.de/newsroom/dark-mode-vs-light-mode-aesthetik-funktionalitaet-und-energieeffizienz-28935`
`https://natively.dev/blog/top-mobile-app-design-trends-2025`

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, CSS Media Queries, moderner Layout-Module wie Flexbox oder Grid sowie Techniken wie Mobile First 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.

`https://www.ionos.at/digitalguide/websites/webdesign/was-bedeutet-responsive-design/`
`https://de.ryte.com/wiki/Responsive_Design/` `https://webstollen.de/responsive-design-od`
`https://kinsta.com/de/blog/responsive-vs-adaptiv/`

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.

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.

Assistive Technologien als Grundlage
Viele Nutzer:innen greifen auf assistive Technologien zurück, die fehlende oder eingeschränkte Fähigkeiten kompensieren. Dazu zählen Screenreader, Vergrößerungstools,externe Eingabegeräte oder Schaltersteuerungen. Besonders Screenreader sind entscheidend, weil sie Inhalte nicht „sehen“, sondern sie anhand von Struktur, Beschriftungen und semantischen Rollen interpretieren. Für die Praxis heißt das, eine Oberfläche kann optisch ansprechend wirken, aber wenn Überschriften fehlen, Buttons nicht sinnvoll beschriftet sind oder die Reihenfolge im Code nicht zur visuellen Logik passt, wird die Bedienung für Screenreader-Nutzer unmöglich.

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.

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, beispielsweiße 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.

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.

Accessibility-Text: Labels und Alt-Text
Ein weiterer zentraler Baustein ist aussagekräftiger Accessibility-Text. Dazu gehören sichtbare Labels, wie Buttontexte, und unsichtbare Beschreibungen wie aria-labels oder contentdescriptions für Icons. Diese Texte sollten kurz, eindeutig und handlungsorientiert sein, weil Screenreader alles vorlesen und lange Formulierungen die Navigation verlangsamen. Für Bilder ist Alt-Text wichtig, wenn das Bild Information trägt: Er soll beschreiben, was relevant ist, ohne unnötige Floskeln. Wenn ein Bild rein dekorativ ist oder bereits durch angrenzenden Text erklärt wird, kann es sinnvoll sein, es für Screenreader zu überspringen, statt redundante Infos vorzulesen.

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. `https://m2.material.io/design/usability/accessibility.html#implementing-accessibility` `https://www.uxmatters.com/mt/archives/`

`2024/05/designing-mobile-apps-with-accessibility-in-mind.php`

3.1.3. UI/UX-Design & Nutzererlebnis

Zielgruppenanalyse & Personas
Eine zielgerichtete Kommunikation und ein passgenaues Angebot setzen voraus, dass klar ist, wer überhaupt angesprochen werden soll und warum diese Personen ein Produkt oder eine Dienstleistung nutzen würden. Genau hier setzt die Zielgruppenanalyse an. Sie ist ein strukturierter Prozess, um potenzielle und bestehende Nutzer besser zu verstehen und daraus konkrete Entscheidungen für Content, Design, Funktionen, Marketing und Produktstrategie abzuleiten. In digitalen Projekten bildet sie damit eine zentrale Grundlage für nutzerorientierte Entwicklung und eine konsistente Markenkommunikation.

Begriff und Bedeutung der Zielgruppe
Eine Zielgruppe beschreibt eine Gruppe von Menschen, die hinsichtlich bestimmter Merkmale wie Alter, Bedürfnisse, Verhalten und Budget ausreichend ähnlich sind, um mit gezielten Maßnahmen angesprochen werden zu können. Die Zielgruppe umfasst dabei nicht nur bereits bestehende Kunden, sondern auch potenzielle Nutzer, die prinzipiell Interesse am Angebot haben könnten.

Ziel und Inhalt der Zielgruppenanalyse
Die Zielgruppenanalyse umfasst alle Aktivitäten, die dazu dienen, diese Gruppe genauer zu beschreiben. Dabei geht es nicht nur um „harte Fakten“ wie demografische Daten, sondern vor allem um die Frage: Welche Bedürfnisse, Motive, Barrieren und Erwartungen bestimmen Entscheidungen und Verhalten? Ziel ist es, ein belastbares Verständnis zu gewinnen, um Maßnahmen nicht nach Bauchgefühl zu planen, sondern zielgerichtet auszurichten.

Personas als Ergebnis der Zielgruppenanalyse
Ein wichtiges Ergebnis dieses Prozesses ist häufig die Erstellung von Personas. Eine Persona ist eine fiktive, aber datenbasierte Modellperson, die typische Merkmale, Ziele und Herausforderungen einer Teilgruppe repräsentiert. Dabei wird unterschieden:

- Buyer Persona: Fokus auf Kauf- und Entscheidungsprozesse (z. B. Budget, Entscheidungskriterien, Einflussfaktoren).
- User Persona: Fokus auf Nutzung und Interaktion (z. B. Ziele bei der Nutzung, Bedienkontext, digitale Kompetenz, typische Aufgaben).

In der Praxis ergänzen sich beide. Während die Buyer Persona stärker Marketing und Vertrieb unterstützt, liefert die User Persona wertvolle Grundlagen für UX, Informationsarchitektur und Feature-Entscheidungen.

Risiken fehlender Zielgruppenanalyse
Ohne Zielgruppenanalyse besteht das Risiko, dass Produkte, Inhalte oder Kommunikationsmaßnahmen an den Bedürfnissen der Nutzer vorbeigehen. Das führt häufig zu ineffizientem

Ressourceneinsatz, geringer Reichweite, schwacher Conversion und langfristig zu sinkender Kundenzufriedenheit. Umgekehrt ermöglicht eine fundierte Analyse, Angebote und Inhalte zielgenau zu gestalten.

Zielgruppenanalyse als iterativer Prozess
Gerade im digitalen Kontext ist außerdem wichtig, dass Zielgruppen sich verändern. Deshalb ist Zielgruppenanalyse kein einmaliger Schritt, sondern ein iterativer Prozess, der regelmäßig überprüft und aktualisiert werden sollte.

Segmentierung von Zielgruppen
Um eine Zielgruppe greifbar zu machen, wird sie anhand verschiedener Kriterien segmentiert. Je mehr Perspektiven kombiniert werden, desto realistischer wird das Bild der Zielgruppe.

B2B- und B2C-Zielgruppen
Ein zentrales Unterscheidungsmerkmal ist dabei, ob sich das Angebot an Unternehmen oder an Endkunden richtet. Im B2B-Bereich sind Entscheidungsprozesse häufig komplexer, da mehrere Entscheidungsträger, formalisierte Prozesse und größere Budgets involviert sind. B2C-Zielgruppen hingegen werden stärker durch individuelle Präferenzen, emotionale Faktoren und Markenbindung beeinflusst.

Geografische, demografische und sozioökonomische Kriterien
Ergänzend dazu spielen geografische Kriterien eine wichtige Rolle, insbesondere bei lokalen oder sprach- und kulturabhängigen Angeboten. Demografische Merkmale sowie sozioökonomische Faktoren wie Bildung, Beruf oder Einkommen tragen dazu bei, die Zielgruppe weiter zu konkretisieren.

Psychografische und verhaltensorientierte Kriterien
Besonders tiefgehende Einblicke liefern psychografische und verhaltensorientierte Kriterien. Dazu zählen unter anderem Werte, Motive, Mediennutzung, Markenpräferenzen sowie das tatsächliche Nutzungs- und Kaufverhalten.

Methoden der Datenerhebung
Zur Datenerhebung kommen unterschiedliche Methoden zum Einsatz, darunter Kundenbefragungen, Social-Media-Monitoring, Webanalysen und die Auswertung von Keyword-Daten. Während quantitative Methoden vor allem aufzeigen, was passiert, liefern qualitative Methoden Erklärungen dafür, warum bestimmtes Verhalten auftritt.

Während Zielgruppen häufig eher abstrakt bleiben, machen Personas diese Informationen handlungsorientiert. Eine Persona bündelt typische Merkmale und ergänzt sie um:

- Ziele (Was will die Person erreichen?)
- Herausforderungen/Schmerzpunkte (Was hindert sie?)
- Entscheidungskriterien (Warum kauft/nutzt sie etwas, oder nicht?)

- Kontext der Nutzung (Wann, wo, mit welchem Gerät?)
- Informations- und Medienverhalten (Wie recherchiert sie?)
- Erwartung an Tonalität, Design, Vertrauen und Service

Wichtig ist: Personas sind keine Fantasiefiguren, sondern sollen auf Daten beruhen. Sie helfen Teams dabei, Entscheidungen zu treffen über themen wie welche Funktionen Priorität haben, welche Inhalte fehlen oder welcher Kommunikationsstil sich am besten eignet. `https://www.appinio.com/de/blog/marktforschung/zielgruppenanalyse` `https://www.webdesign-journal.de/zielgruppenanalyse/`

User Journey
Eine User Journey beschreibt den gesamten Weg, den Nutzende durchlaufen, um mit einem Produkt oder Service ein bestimmtes Ziel zu erreichen. Dabei werden alle relevanten Touchpoints, Interaktionen sowie die Emotionen und Entscheidungen der Nutzenden berücksichtigt. Ziel einer User Journey ist es, das Nutzungserlebnis ganzheitlich zu verstehen und Optimierungspotenziale sichtbar zu machen, um Produkte langfristig nutzerfreundlicher und besser zu gestalten. Ein optimiertes Nutzererlebnis stärkt nicht nur die Zufriedenheit, sondern auch die emotionale Bindung zur Marke. Das Ergebnis sind loyale Nutzende, die das Produkt weiterempfehlen und langfristig nutzen.

Eine User Journey Map beantwortet unter anderem folgende zentrale Fragen:

- Welche Schritte durchlaufen Nutzende vom ersten Kontakt bis zur Zielerreichung?
- Wie interagieren sie aktuell mit dem Produkt oder Service und wie könnte diese Interaktion idealerweise aussehen?
- Welche Faktoren wie Emotionen, Budget, Werte, Zeitdruck oder Informationslage beeinflussen ihre Entscheidungen?
- Wo treten Schwierigkeiten oder Frustrationen auf?
- Wie kann das Produkt Nutzende besser unterstützen und effizienter zum Ziel führen?
- An welchen Stellen lässt sich das Nutzererlebnis optimieren, um eine langfristige Kundenbindung aufzubauen?
- Wie unterscheidet sich die Nutzung zwischen verschiedenen Zielgruppen?

Unabhängig vom Produkt lassen sich User Journeys typischerweise in mehrere Phasen gliedern:

- Aufmerksamkeit – Nutzende erkennen ein Bedürfnis oder Problem
- Bewertung – verschiedene Optionen werden verglichen
- Überlegung – eine konkrete Entscheidung wird vorbereitet
- Einkauf/Nutzung – Kauf oder aktive Nutzung des Produkts
- Bindung – Nachkaufphase, Feedback, Wiederkehr und Loyalität

Diese Phasen sind nicht strikt linear, sondern können sich wiederholen oder überschneiden. Eine positive Erfahrung in der Bindungsphase kann Nutzende erneut in die Aufmerksamkeits-Phase bringen, ein stabiler Kreislauf, der für Unternehmen langfristig Gewinn bringt.

Während der User Journey werden Nutzende von zahlreichen Faktoren beeinflusst, darunter persönliche Werte, Budget, Zeit, Benutzerfreundlichkeit der Platform und Servicequalität. Sogennante Pain Points, beschreiben stellen, an denen ein User nicht mehr eindeutig weiß was zu tun ist, sie entstehen vor allem dann, wenn Informationen fehlen, Prozesse unübersichtlich sind oder Erwartungen nicht erfüllt werden. Genau hier setzt User Journey Mapping an. Diese Problemstellen werden sichtbar gemacht und gezielt adressiert.

Als Ergebnis entsteht eine grafisch aufbereitete User Journey Map, die alle Phasen, Schritte, Touchpoints und Emotionen der Nutzenden darstellt. Ergänzt wird diese durch zusätzliche Informationen wie Nutzerzitate, genutzte Kanäle oder Endgeräte. Interaktive User Journey Maps ermöglichen es zudem, Inhalte flexibel ein- und auszublenden und die Analyse gezielt zu vertiefen. `https://www.usability.de/leistungen/methoden/user-journey-mapping.html` `https://mailchimp.com/de/resources/user-journey/` `https://piwikpro.de/glossar/user-journey/`

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.

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.

Ein hilfreiches Analogbeispiel ist die Bibliothek: Kategorien, Beschilderungen und Kataloge sorgen dafür, dass Besucher das gesuchte Buch effizient finden. Eine gute Informationsarchitektur erfüllt dieselbe Aufgabe im Digitalen, sie organisiert Information so, dass Suchwege kurz bleiben und Orientierung leichtfällt.

Auf Basis etablierter Informationsarchitektur-Modelle lässt sie sich in vier zentrale Komponenten gliedern:

- 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 Websiten relevant.

Ein gutes Informationsarchitektur-Design achtet auf folgende Prinzipien:

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

`https://www.eresult.de/blog/ux-design/blog-informationsarchitektur-ux/` `https://b13.com/de/blog/informationsarchitektur-in-ux-best-practices`

Wireframes & Prototypen
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.

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.

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.

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.

Zusammenspiel im Designprozess
Wireframing und Prototyping sind keine konkurrierenden Phasen, sondern ergänzen sich. In der Regel beginnt der Designprozess mit Wireframes, die als Fundament dienen. 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.

Abgrenzung zu Mockups
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. `https://miro.com/de/wireframing/wireframe-vs-prototyp/#high-fidelity-wireframe-vs.-prototyp` `https://www.justinmind.com/de/wireframe/unterschied-wireframe-vs-prototyp-vs-mockup` `https://www.popwebdesign.de/popart_blog/de/2017/06/wireframe-vs-prototype-was-ist-der`

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.

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.

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.

Weißraum als Strukturprinzip
Ein oft unterschätztes, aber wesentliches Element ist der Weißraum, auch negativer Raum genannt. Er sorgt für Struktur, Klarheit und visuelles Gleichgewicht. Durch gezielten Einsatz von Weißraum werden Inhalte voneinander abgegrenzt, wichtige Elemente hervorgehoben und die kognitive Belastung reduziert. Dadurch verbessert sich die Nutzerführung erheblich.

Evaluation und Optimierung im Designprozess
Moderne visuelle Designs werden zunehmend daten- und nutzerorientiert entwickelt. Mithilfe von Benutzeranalysen, Usability-Tests und Rapid-Prototyping-Methoden lassen sich Designentscheidungen überprüfen und optimieren. Die Wirkung visueller Gestaltung ist somit messbar und kann gezielt zur Verbesserung der Nutzererfahrung und Markenwahrnehmung eingesetzt werden. `https://toolmaster.ch/visuelles-design-was-ist-visuelles-design/` `https://www.studysmarter.de/schule/kunst/grafikdesign-kunst/visual-design/` `https://www.studysmarter.de/schule/kunst/grafikdesign-kunst/visual-design/`

3.1.4. Designentscheidungen im Hinblick auf Monetarisierung

- Einfluss von UI/UX auf die Zahlungsbereitschaft der Nutzer
- Sichtbare, aber nicht aufdringliche Platzierung von Call-to-Actions
- Dezente Integration von Bezahl- und Werbeelementen
- Bewusste Vermeidung von manipulativen Dark Patterns

3.2. Monetarisierung von mobilen Apps

3.2.1. Grundlagen & Marktüberblick

Entwicklung des mobilen App-Marktes
Der mobile App-Markt wächst kontinuierlich und bietet vielfältige Monetarisierungsmöglichkeiten.

App-Ökosysteme
Die wichtigsten Distributionsplattformen für mobile Apps sind:

- Apple App Store
- Google Play Store

Nutzerverhalten & Marktstatistiken
Marktanalysen zeigen, dass Nutzer zunehmend bereit sind, für digitale Inhalte und Zusatzfunktionen zu bezahlen.

Geschäftsmodelle digitaler Produkte
Digitale Produkte ermöglichen unterschiedliche Erlösmodelle wie Einmalkäufe, Abonnements oder In-App-Käufe.

3.2.2. Werbung als Einnahmequelle

Arten von Werbung in mobilen Apps
In mobilen Apps kommen verschiedene Werbeformen wie Banner-, Interstitial- oder Videoanzeigen zum Einsatz.

Implementierung von Werbung
Werbung kann über externe Werbenetzwerke technisch einfach in Apps integriert werden.

Werbenetzwerke
Häufig genutzte Werbenetzwerke sind:

- Google AdMob
- Smaato
- Meta Audience Network
- AppLovin

Auswirkungen auf die User Experience
Eine ausgewogene Platzierung von Werbung ist entscheidend, um Einnahmen zu erzielen, ohne die Nutzererfahrung negativ zu beeinflussen.

3.2.3. Sponsoring & Partnerschaften

Grundprinzip von App-Sponsoring
Unternehmen können als Sponsoren innerhalb der App auftreten und so gezielt Zielgruppen erreichen.

Kooperationen mit Tourismusverbänden
Tourismusverbände eignen sich besonders für regionale oder themenspezifische Anwendungen.

Hotel- & Buchungsplattformen
Mögliche Kooperationspartner sind:

- Trivago
- Check24
- Ab-in-den-Urlaub

Wirtschaftliches Potenzial
Partnerschaften können eine langfristige und stabile Einnahmequelle darstellen.

3.2.4. Kostenpflichtige App-Modelle

Einmaliger Kauf
Die App wird gegen eine einmalige Gebühr angeboten.

Abonnement-Modelle
Nutzer zahlen regelmäßig für den Zugriff auf zusätzliche Inhalte oder Funktionen.

Zahlungsbereitschaft der Nutzer
Die Zahlungsbereitschaft hängt stark vom wahrgenommenen Mehrwert der App ab.

3.2.5. In-App-Käufe

Grundlagen von In-App-Purchases
Zusätzliche Inhalte oder Funktionen können direkt innerhalb der App erworben werden.

Personalisierung
Beispiele für personalisierte Inhalte sind:

- Profilbilder
- Banner

Premium-Funktionen
Bestimmte Funktionen sind ausschließlich zahlenden Nutzern vorbehalten.

Abo-Modelle
Monatliche Abonnements ermöglichen den Zugang zu erweiterten Features.

3.2.6. Spenden & freiwillige Unterstützung

Spendenmodelle in Apps
Nutzer haben die Möglichkeit, die App freiwillig finanziell zu unterstützen.

Transparenz & Vertrauen
Eine offene Kommunikation über die Verwendung der Spenden stärkt das Vertrauen der Nutzer.

3.2.7. Vergleich & Bewertung der Monetarisierungsstrategien

Wirtschaftliches Potenzial
Die Monetarisierungsstrategien unterscheiden sich deutlich hinsichtlich ihres Ertragspotenzials.

Nutzerakzeptanz
Die Akzeptanz der Nutzer ist ein zentraler Erfolgsfaktor für jedes Monetarisierungsmodell.

Geeignete Strategie für die entwickelte App
Abschließend wird die am besten geeignete Monetarisierungsstrategie für die entwickelte App bewertet.

3.3. Laurenz Ausarbeitung

3.3.1. Mobile App Entwicklung

3.3.1.1. Native vs. Cross-Platform Entwicklung

Quellenangabe: [https://www.denovo.at/blog/native-vs-cross-plattform-app-welcher-weg-ist-der-richtige-fuer-ihre-idee]

Es existieren verschiedene technische Herangehensweisen bei der Entwicklung mobiler Apps, um Anwendungen für mobile Endgeräte wie Smartphones und Tablets zu realisieren. Dabei gibt es einen wesentlichen Unterschied zwischen der Entwicklung nativer Apps und der Cross-Plattform-Entwicklung. Beide Optionen weisen Unterschiede hinsichtlich der Architektur, des Entwicklungsprozesses und der technischen Realisierung auf.

Unter nativer App-Entwicklung versteht man die Entwicklung und Umsetzung von Anwendungen, die speziell für ein bestimmtes Betriebssystem programmiert worden sind. Beispielsweiße sind das für iOS-Anwendungen, die auf Swift basieren, während Android-Applikationen typischerweise in Kotlin entwickelt werden. Die Anwendung wird direkt für dieses bestimmte System entwickelt und ist mit keinem anderen kompatibel.

Ein wesentliches Merkmal nativer Anwendungen ist der direkte Zugriff auf systemnahe Funktionen und Hardware-Komponenten des Endgeräts, wie Kamera, GPS oder Sensoren. Darüber hinaus orientieren sich native Apps stark an den Design- und Interaktionsrichtlinien des jeweiligen Betriebssystems, wodurch sie sich nahtlos in dessen Benutzeroberfläche einfügen.

Im Gegensatz dazu steht die Cross-Platform-App-Entwicklung, bei der Anwendungen mit einer gemeinsamen Codebasis für mehrere Betriebssysteme entwickelt werden. Ziel dieses Ansatzes ist es, den Entwicklungsaufwand zu reduzieren, indem große Teile des Codes plattformübergreifend wiederverwendet werden. Die Ausführung erfolgt anschließend auf verschiedenen Betriebssystemen, meist iOS und Android.

Cross-Platform-Frameworks abstrahieren die zugrunde liegenden Betriebssysteme und stellen einheitliche Schnittstellen zur Verfügung. Dadurch können Entwicklerinnen und Entwickler eine Anwendung erstellen, die auf mehreren Plattformen lauffähig ist, ohne für jedes

Betriebssystem eine vollständig eigene Implementierung zu benötigen. Dennoch erfolgt die Interaktion mit gerätespezifischen Funktionen häufig über zusätzliche Abstraktionsschichten oder spezielle Erweiterungen.

3.3.1.2. React Native Framework

Quellenangabe: [https://brainhub.eu/de/library/was-ist-react-native]

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.

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.

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.

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.

3.3.2. Backend-Technologien

3.3.2.1. APIs

Quellenangabe: [https://www.talend.com/de/resources/was-ist-eine-api/]

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.

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.

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.

3.3.2.2. Datenbanken

Quellenangabe: [https://www.ibm.com/think/topics/database]

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.

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.

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.

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.

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

Quellenangabe: [https://medium.com/@anshuldevx/overview-of-supabase-backend-as-a-service-platform-e192da9a369c]

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.

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:

- **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.

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.

3.3.3. Hosting- und Infrastrukturmodelle

3.3.3.1. On-Premise, Cloud und Hybrid-Architekturen

Quellenangabe: https://hws-gruppe.de/on-premises-in-die-cloud-oder-doch-hybrid/
https://www.ibm.com/de-de/think/topics/hybrid-cloud-architecture

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.

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

Quellenangabe: https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-iaas
https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-paas
https://www.ibm.com/de-de/think/topics/faas
https://www.redhat.com/de/topics/cloud-native-apps/what-is-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.

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.

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.

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.

3.3.3.3. Relationale Datenbanken

Quellenangabe: https://www.oracle.com/de/database/what-is-a-relational-database/

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.

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

Quellenangabe: https://db-engines.com/de/article/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.

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

Quellenangabe: https://www.medienpalast.net/glossar/begriff/real-time-data/

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.

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

Quellenangabe: https://www.akamai.com/de/glossary/what-is-horizontal-scaling-vs-vertical-scaling

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.

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.

3.3.3.7. Autoscaling und Lastverteilung

Quellenangabe: https://teamtakt.de/devops-tool-teil-1-die-bedeutung-von-lastverteilung-cdns-und-automatischem-skalieren-in-iaas/

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.

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.

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

3.3.3.8. Latenzoptimierung und Durchsatzsteigerung

Quellenangabe: https://www.studysmarter.de/schule/informatik/technische-informatik/latenz-und-durchsatz/

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.

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.

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.

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.

3.3.3.9. Auth-Systeme

Quellenangabe: https://www.it-schulungen.com/wir-ueber-uns/wissensblog/welche-modernen-authentifizierungsmoeglichkeiten-gibt-es.html

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.

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.

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.

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.

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.

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.

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.

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.

3.3.3.10. Verschlüsselung

Quellenangabe: https://www.kaspersky.de/resource-center/definitions/encryption

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.

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.

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.

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.

3.3.3.11. Datenschutz

Quellenangabe: https://www.onlinesicherheit.gv.at/Services/Technologie-Schwerpunkte/Mobile-Apps/Mobile-Apps-und-Informationspflichten.html

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.

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.

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.

4. Dokumentation der Implementierung

4.1. Grundlagen der Implementierung

4.1.1. Systemumgebung

Die entwickelte Anwendung ist ein mobiles Spiel für iOS, umgesetzt mit React Native und TypeScript. Für die Entwicklung wurde Visual Studio Code unter macOS verwendet. Die Anwendung kann über ein Apple-Developer-Konto direkt auf ein iPhone installiert werden und ist für eine zukünftige Veröffentlichung im Apple App Store vorgesehen.

4.1.2. Verwendete Technologien

- **React Native** – Framework zur Entwicklung plattformübergreifender mobiler Anwendungen.
- **TypeScript** – Typsichere Erweiterung von JavaScript zur Erhöhung der Codequalität.
- **Supabase (PostgreSQL)** – Backend-as-a-Service zur Bereitstellung einer REST-API, Authentifizierung und Datenbank.
- **Google Street View API** – Externe API zur Integration von Street-View-Bildern in das Spiel.
- **GitHub** – Versionsverwaltung und Projektmanagement.

4.1.3. Architekturüberblick

Die Anwendung nutzt eine klassische Client–Server-Struktur. Der Client (React Native App) kommuniziert über die Supabase-REST-API mit einer PostgreSQL-Datenbank. Zusätzlich werden externe Anfragen an die Google Street View API gesendet.

Administrator- und Benutzerrollen sind implementiert, wobei Administratoren erweiterte Berechtigungen besitzen (Freigeben von globalen Nachrichten, Einsehen von Meldungen).

4.2. Implementierung der Funktionen

4.2.1. Login und Registrierung

Die Authentifizierung erfolgt über Supabase Auth. Es werden Funktionen zur Registrierung, Anmeldung und Passwortverwaltung bereitgestellt. Alle Benutzerdaten werden sicher in der PostgreSQL-Datenbank gespeichert.

4.2.2. Benutzerverwaltung

Das System unterstützt zwei Rollen:

- **Benutzer** – reguläre Spieler.
- **Premium** – reguläre Spieler ohne Werbung.
- **Developer** – erweiterte Rechte, z. B. Verwaltung von Meldungen und Veröffentlichung globaler Nachrichten zur Testung der App.
- **Administrator** – erweiterte Rechte, z. B. Verwaltung von Meldungen und Veröffentlichung globaler Nachrichten.

Spieler können andere melden; Meldungen werden in Supabase gespeichert und durch Administratoren einsehbar.

4.2.3. Einzelspieler-Modus

Der Einzelspielermodus nutzt die Google Street View API zur Anzeige zufälliger Standorte. Der Spieler gibt eine Vermutung ab und erhält Punkte basierend auf der Distanz zum tatsächlichen Ort.

4.2.4. Mehrspieler-Modus

Spieler können eine Lobby erstellen, beitreten oder suchen. Die Kommunikation erfolgt in Echtzeit über Supabase Channels. Jeder Spieler sieht dasselbe Street-View-Bild; das System wertet anschließend die Ergebnisse aus und führt eine Rangliste.

4.2.5. Bestenliste

Die globalen Highscores werden in der Datenbank gespeichert. Ein Ranking wird dynamisch aus den Spielergebnissen generiert.

4.2.6. Tägliches Spiel

Jeden Tag steht allen Nutzern dasselbe Bild zur Verfügung. Die Ergebnisse werden global verglichen und in einer separaten Tabelle gespeichert.

4.2.7. Österreichweite Spielmodi

Es existieren zwei Varianten:

- **Ganz Österreich** – zufällige Orte im gesamten Bundesgebiet.
- **Spezifische Bundesländer** – Einschränkung auf einzelne Regionen.

4.2.8. Chat mit Freunden

Spieler können miteinander chatten. Nachrichten werden über Supabase Realtime synchronisiert und persistent gespeichert.

4.2.9. Freundesliste und Anfragen

Es besteht die Möglichkeit:

- nach Freunden zu suchen,
- Freundschaftsanfragen zu senden und zu akzeptieren,
- eine bestehende Liste von Freunden zu verwalten.

4.2.10. Profilbilder und Shop-System

Spieler können Profilbilder „kaufen“. Der Kaufvorgang ist derzeit simuliert und dient als Platzhalter für ein zukünftiges Zahlungssystem.

4.3. Datenhaltung

Die Datenbank basiert auf PostgreSQL mit Supabase als Service. Tabellen umfassen u. a.:

- Benutzer
- Freundesbeziehungen
- Chats und Nachrichten
- Meldungen
- Ergebnisse (Einzelspieler/Mehrspieler)
- tägliche Spiele
- globale Nachrichten

4.4. Deployment

Während der Entwicklungsphase wird die App lokal über ein Apple-Developer-Konto auf iOS-Geräten ausgeführt. Eine zukünftige Veröffentlichung im Apple App Store ist vorgesehen.

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] abc: *DB-Engine Ranking*, März 2016. Online in Internet: URL: `http://db-engines.com/de/ranking`.

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