

Nochba – App für Nachbarschaftshilfe

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Informatik

Eingereicht von:

Arsham Edalatkhah
Sandin Habibovic
Martin Hausleitner

Betreuer:

Rainer Stropek

Projektpartner:

IKT Linz GmbH - Open Common Linz

Leonding, März 2023

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 bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2022

S. Schwammal & S. Schwammal

Abstract

Brief summary of our amazing work. In English. This is the only time we have to include a picture within the text. The picture should somehow represent your thesis. This is untypical for scientific work but required by the powers that are. Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.



Zusammenfassung

Zusammenfassung unserer genialen Arbeit. Auf Deutsch. Das ist das einzige Mal, dass eine Grafik in den Textfluss eingebunden wird. Die gewählte Grafik soll irgendwie eure Arbeit repräsentieren. Das ist ungewöhnlich für eine wissenschaftliche Arbeit aber eine Anforderung der Obrigkeit. *Bitte auf keinen Fall mit der Zusammenfassung verwechseln, die den Abschluss der Arbeit bildet!* Suspendisse vel felis.

Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.



Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

1.2 Aufgabenstellung

1.3 Zusammenfassung des Projektergebnisse

1.4 Projektvorgehensmodell

2 Technologien

2.1 Technologieevaluierung

Im Rahmen dieser Diplomarbeit wurden umfangreiche Anforderungen an das Frontend und Backend der Anwendung gestellt, um eine hochwertige und funktionale Anwendung zu entwickeln. Für das Frontend wurden iOS- und Android-Apps gefordert, wobei die Möglichkeit einer zukünftigen Web-App berücksichtigt werden sollte. Das Backend sollte eine Echtzeit-Datenbank für den Chat und eine Datenbank mit Geofunktionen für den Radios-Filter bereitstellen. Dabei war es wichtig, dass das Backend einfach zu implementieren ist und eine gute SDK-Unterstützung für das Frontend bietet.

Neben diesen technischen Anforderungen war auch die Umsetzung der Business-Logik ein wichtiger Aspekt. Hierbei wurden Aspekte wie die Registrierung und Verifizierung von Nutzern berücksichtigt, um eine sichere und benutzerfreundliche Anwendung zu gewährleisten. Darüber hinaus war es wichtig, dass die Anwendung mit neuen und effizienten Technologien entwickelt wird, um eine schnelle und agile Anpassung an die sich ändernden Anforderungen des Marktes und der Nutzer zu ermöglichen.

2.1.1 Frontend

Nach ausführlicher Recherche wurden die folgenden plattformübergreifenden Frameworks in Betracht gezogen[?]: Flutter, React Native, Ionic und Xamarin. Jede Plattform hat ihre Vor- und Nachteile.

Nach langer Recherche und Evaluierung aller Faktoren wurde Flutter als Plattform für das Frontend ausgewählt. Insbesondere die schnelle Entwicklung und das einfache Debugging sowie die Integration in VS Code waren entscheidende Faktoren. Jeder aus unserem Team hat eine eigene Flutter Test App programmiert [?] um die Entscheidung zu festigen.

Insgesamt waren wir mit unserer Entscheidung, Flutter zu verwenden, sehr zufrieden. Während der Entwicklungszeit stellte sich heraus, dass die Klammernverwendung in Flutter am Anfang etwas ungewohnt war, aber dank der hervorragenden Unterstützung

durch die Flutter-Community und der umfangreichen Dokumentation auf Stack Overflow konnten wir alle Herausforderungen bewältigen. Außerdem wurde die Entwicklung durch die ständigen Verbesserungen und Updates von Flutter, wie der Einführung der Impeller-Rendering-Engine, beschleunigt. [?]

Insgesamt war die Wahl von Flutter für das Frontend der App eine gute Entscheidung.

2.1.2 Backend

Für unser Projekt kamen drei Technologien in Frage: Supabase [?], Firebase[?] und Appwrite[?]. Sie alle bieten SDKs für Flutter und verfügen über eine benutzerfreundliche Authentifizierungsschicht. Darüber hinaus verfügen sie alle über eine Datenbank, die Geoqueries für das Hauptfeature des Beitragsradius unterstützt, sowie einen Speicherplatz für Beitragsfotos.

Allerdings bietet Supabase zum Zeitpunkt der Evaluierung im Februar 2022 noch keine Cloud-Funktionen wie Firebase und Appwrite, was es schwierig macht, eine robuste Business-Logik umzusetzen. Obwohl Supabase am 1. April 2022 experimentell Edge Functions eingeführt hat, ist dies für eine Produktions-App nicht empfehlenswert.

Daher wurden Appwrite und Firebase als die beiden besten Optionen identifiziert. Firebase bietet bessere Flutter-SDKs und verfügt über mehr Features, einschließlich der Unterstützung von Google für Flutter, weshalb wir uns schließlich für Firebase entschieden haben.

In der folgenden Tabelle sind die Vor- und Nachteile von Appwrite und Firebase aufgeführt:

Insgesamt bietet Firebase aufgrund der umfangreichen Features und der einfacheren Integration mit anderen Google-Tools Vorteile gegenüber Appwrite.

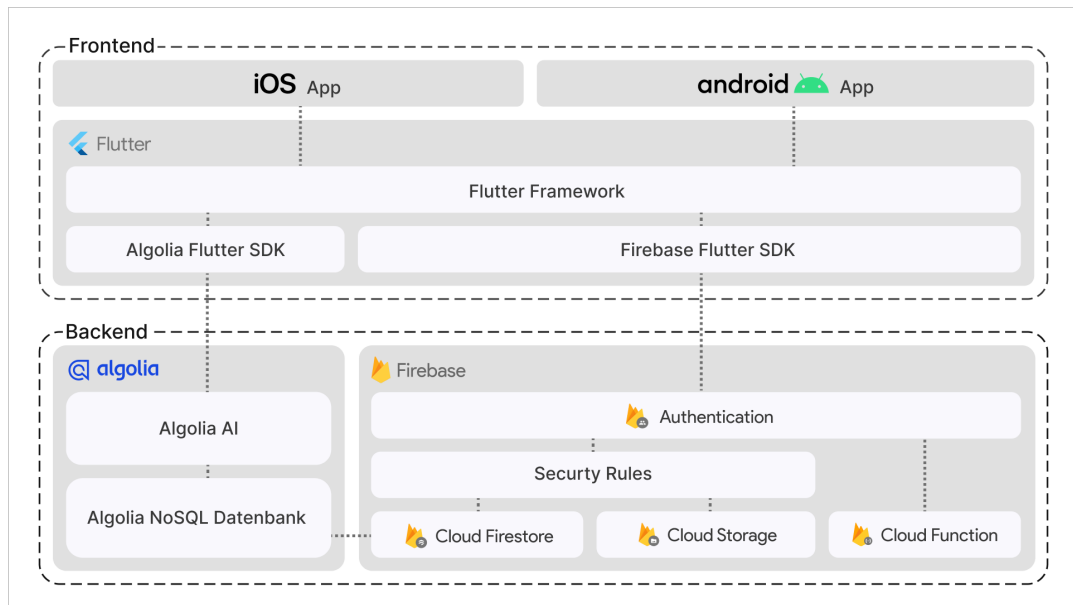
2.2 Fazit

Die Wahl von Flutter als Frontend-Plattform und Firebase als Backend-Plattform erwies sich als gute Entscheidung für die Anforderungen dieser Diplomarbeit. Die Einfachheit der Syntax, das schnelle Debugging und die umfangreiche Dokumentation und Unterstützung durch die Community machten Flutter zu einer einfachen und

effizienten Plattform für die Entwicklung der mobilen Apps. Firebase hat eine gute Lösung für die Echtzeit-Datenbank und die Geodatenbank und eine einfache Integration mit Flutter.

Insgesamt haben die gewählten Technologien dazu beigetragen, dass die Entwicklung der Anwendung schnell und effizient verlief und die Anforderungen der Diplomarbeit erfüllt wurden.

3 Systemarchitektur



3.1 Flutter

Hier kommt die Beschreibung der Technologie und deren Vorteile/Nachteile, auf Basis von wissenschaftlichen Studien und Erfahrungen aus der Praxis.

Quellen:

<https://docs.flutter.dev/resources/architectural-overview>

<https://flutter.dev/docs/resources/technical-overview>, <https://pub.dev/packages/flutter>

3.2 Firebase

Firebase[?] ist eine von Google entwickelte Plattform für die Entwicklung von Web- und mobilen Anwendungen. Die Plattform bietet eine Vielzahl von Tools und Diensten, die es Entwicklern ermöglichen, schnell und einfach skalierbare Anwendungen zu erstellen.

Die Systemarchitektur von Firebase basiert auf der Cloud-Computing-Technologie, bei der die Anwendungslogik und Daten in der Cloud gehostet werden. Dies bedeutet, dass Entwickler keine physischen Server oder Infrastrukturen verwalten müssen, um ihre Anwendungen zu betreiben. Stattdessen können sie sich auf die Entwicklung der Anwendungslogik konzentrieren und die Firebase-Plattform übernimmt den Rest.

Firebase bietet auch eine Echtzeit-Datenbank, die es Entwicklern ermöglicht, Daten in Echtzeit zwischen ihren Anwendungen zu synchronisieren. Darüber hinaus bietet Firebase eine Vielzahl von Tools und Diensten, darunter Authentifizierung, Benachrichtigungen, Hosting, Speicherung und vieles mehr.

Dank dieser Systemarchitektur und der bereitgestellten Tools und Dienste können Entwickler mit Firebase schnell und einfach skalierbare Anwendungen erstellen und betreiben.

3.2.1 Firebase Authentication

Beschreibung der Firebase Authentication Technologie, inklusive Sicherheitsregeln und Best Practices.

Quellen:

<https://firebase.google.com/docs/auth>

<https://medium.com/flutter-community/firebase-authentication-in-flutter-752d14209a8a>

Security Rules

In unserer App ist die Sicherheit ein sehr wichtiger Faktor, um die Vertraulichkeit und Integrität der Daten und der Nutzer zu gewährleisten. Firebase bietet für seine Cloud-basierten Datenbank- und Speicherlösungen - Cloud Firestore und Cloud Storage - die sogenannten Security Rules[?][?][?], um den Zugriff auf die Daten und Ressourcen zu kontrollieren. Diese Regeln definieren, wer auf welche Art und Weise auf welche Daten zugreifen darf.

Die Security Rules für Cloud Firestore und Cloud Storage sind in einer eigenen Sprache geschrieben und werden serverseitig auf Firebase-Servern ausgeführt. Die Syntax basiert auf einer ähnlichen Struktur wie JSON und erlaubt komplexe Abfragen. Die Regeln können für eine bestimmte Sammlung oder einen bestimmten Pfad definiert werden und erlauben es, bestimmte Bedingungen für Lese- oder Schreibzugriffe zu definieren.

Listing 1: Security Rules Beispiel

```
1      match /posts/{postId} {  
2          allow read: if request.auth.uid != null;  
3          allow write: if request.auth.uid == resource.data.author;  
4      }
```

Diese Regel definiert, dass jeder Nutzer Lesezugriff auf alle Posts hat, aber nur der Autor des Posts ihn auch ändern darf.

Insgesamt bieten die Security Rules für Cloud Firestore und Cloud Storage eine leistungsstarke und flexible Möglichkeit, die Zugriffsrechte auf die Daten und Ressourcen in einer Firebase-App zu steuern. Mit Hilfe der Security Rules können wir sicherstellen, dass die Daten ihrer Nutzer geschützt und nur von berechtigten Personen abgerufen oder geändert werden können.

3.2.2 Cloud Firestore

Vorstellung der Datenbanktechnologie Firestore

Quellen:

<https://firebase.google.com/docs/firestore>

<https://medium.com/flutter-community/firebase-cloud-firestore-in-flutter-26c6e8c6f90c>

Datenmodell

Hier kommt die Erklärung des Datenmodells in Firebase Firestore und dessen Auswirkungen auf die App-Architektur.

Quellen:

<https://firebase.google.com/docs/firestore/data-model>

<https://www.raywenderlich.com/6628345-cloud-firestore-for-flutter-getting-started>

Weitere wichtige punkte:

- Präsentation des eigenen Datenmodells in der Arbeit.
- Diagramme werden verwendet, um das Modell zu präsentieren und Entscheidungen zu erläutern und zu begründen.
- Anforderungen der App-Architektur werden dabei berücksichtigt werden.
- Performance, Skalierbarkeit und Strukturierung können thematisiert werden.

- Zur Veranschaulichung unserer eigenen Datenmodell-Entwicklung kann auf ein Beispiel-Datenmodell-Diagramm auf der offiziellen Firebase-Website verwiesen werden, welches uns als Orientierungshilfe diene.

Beispiel-Datenmodell-Diagramm Quelle:

https://firebase.google.com/docs/firestore/data-model#structure_your_data

3.2.3 Cloud Storage

Beschreibung der Cloud Storage Technologie in Firebase und deren Einsatz in der App.

Quellen:

<https://firebase.google.com/docs/storage>

<https://medium.com/flutter-community/firebase-cloud-storage-in-flutter-flutter-an-firebase-tutorial-c5de7835c6cd>

3.2.4 Firebase Cloud Functions

Firebase Cloud Functions[?] ermöglichen es uns, unsere Business-Logik wie beispielsweise den Registrierungsprozess und viele andere Logikvorgänge einfach und effektiv in der Cloud abzubilden. Wir haben uns entschieden, die Funktionen in TypeScript zu schreiben, da es uns viele Vorteile bietet, wie zum Beispiel eine statische Typisierung, verbesserte IDE-Unterstützung und eine erhöhte Code-Lesbarkeit.

Der Registrierungsprozess ist ein gutes Beispiel dafür, wie wir Firebase Cloud Functions einsetzen können. Anstatt eine monolithische Anwendung zu erstellen, die alles in einem einzigen Server handhabt, können wir die Logik in kleinere Funktionen aufteilen, die jeweils eine spezifische Aufgabe erfüllen.

Firebase Cloud Functions bietet viele Vorteile, wie zum Beispiel die Möglichkeit, die Funktionen einfach zu skalieren und automatisch zu verteilen, um hohe Lasten zu bewältigen. Außerdem können wir die Funktionen einfach testen und debuggen, indem wir lokale Emulatoren verwenden, bevor wir sie in der Cloud bereitstellen.

Insgesamt sind Firebase Cloud Functions eine großartige Möglichkeit, um unsere Business-Logik in der Cloud abzubilden und unsere Anwendung zu skalieren und

zu verbessern. Durch die Verwendung von TypeScript können wir sicherstellen, dass unser Code sauber und robust bleibt und leicht gewartet werden kann.

3.3 Algolia Search

Vorstellung der Algolia Search Technologie und deren Integration in die App-Architektur.

Algolia AI

Quellen:

<https://www.algolia.com/doc/>

<https://www.algolia.com/doc/guides/sending-and-managing-data/send-and-update-your-data/tutorials/firebase-algolia/>

Firestore Cloud Function

Beschreibung der Firestore Cloud Functions und deren Rolle in der Algolia Integration.

Quellen:

<https://firebase.google.com/docs/functions>

<https://www.algolia.com/doc/guides/sending-and-managing-data/send-and-update-your-data/tutorials/firebase-algolia/>

4 Umsetzung

4.1 Continuous Integration/Delivery

4.1.1 GitHub Actions

allg. actions warum kosten

Build IOS

Build Android

4.1.2 Fastlane

Build Number increment

4.1.3 Firebase App Distribution

4.2 Mobile Anwendung

4.2.1 Dateistruktur

In Flutter gibt es keine fixe Dateistruktur für eine App, man kann seine Struktur also selbst überlegen und gestalten. Im Folgenden beschreibe ich, wie wir unsere Dateistruktur für eine Flutter-App aufgebaut haben.

In Flutter gibt es keine feste Dateistruktur, stattdessen kann man die Struktur der Dateien und Ordner selbst bestimmen. Für unser Flutter-Projekt haben wir uns für eine Struktur entschieden, die sich an bewährten Praktiken orientiert.

Unsere Dateistruktur sieht wie folgt aus:

- **logic** - Hier befindet sich die Geschäftslogik der App, einschließlich der Firestore-Cloud-Funktionen und Repositories, die API-Aufrufe ausführen.
- **pages** - Hier werden Widgets entworfen, die jeweils eine Seite der App darstellen.
- **routes** - Hier werden die Routen definiert, die tiefere Links ermöglichen.

- **shared** - Hier werden UI-Widgets wie Buttons oder andere Widgets gespeichert, die oft wiederverwendet werden.
- **views** - Hier befinden sich Ansichten, die von mehreren Seiten der App verwendet werden können.

Im Nachhinein hätten wir die Dateistruktur anders gestaltet, z.B. hätten wir das UI als eigenes Package definiert und die pages und views besser unterteilt.

4.2.2 State Management

In Flutter gibt es verschiedene Möglichkeiten[?][?], um mit dem State Management umzugehen. State Management bezieht sich auf die Art und Weise, wie Daten innerhalb einer App verwaltet werden. In jeder App gibt es bestimmte Daten, die von verschiedenen Komponenten und Widgets verwendet werden und sich im Laufe der Zeit ändern können. State Management bezieht sich auf die Methoden, die verwendet werden, um diese Daten innerhalb der App zu verwalten und zu aktualisieren.

GetX

GetX verwendet ein reaktives Ansatz zur Verwaltung des Zustands, was bedeutet, dass Änderungen im Zustand automatisch die UI aktualisieren, ohne dass der Entwickler manuell Code schreiben muss, um diese Aktualisierungen durchzuführen. Dies spart viel Entwicklungszeit und macht es einfach, auf Benutzerinteraktionen zu reagieren.

Mit GetX können wir auch eine einheitliche Datenquelle haben, auf die alle Komponenten zugreifen können, was die Wartung und Erweiterung der Anwendung erleichtert. Darüber hinaus bietet GetX auch eine einfache Möglichkeit, Abhängigkeiten zu verwalten und Zustandsinformationen zwischen Bildschirmen zu teilen.

Insgesamt hat uns die Verwendung von GetX im Flutter-Framework sehr geholfen, eine effektive und skalierbare Anwendung zu erstellen, die auf die Bedürfnisse unserer Benutzer abgestimmt ist. `getx controller service etc...`

4.2.3 Authentifizierung

Anmelde Flow

Diagram erklärung screenshots

Registrierungs Flow

Diagram erklärung screenshots

Firebase Authentifizierung

allg.

4.2.4 Feed

foto aufbau

4.2.5 Beiträge

Beiträge sind das Hauptkommunikationsmittel auf der App. Jedem Beitrag muss ein Titel, eine Beschreibung und eine Reichweite, unter der, der Beitrag sichtbar ist, angegeben werden. Weiters ist es möglich einem Beitrag ein Bild und Tags anzuhängen.

Kategorien

Um Beiträge besser zuordnen zu können, muss der User den Beitrag vor dem Veröffentlichen in eine bestimmte Kategorie einteilen. Diese Kategorien ermöglichen es Usern, die Art Ihrer Anfrage ihm vorhinein besser zu spezifizieren und die Suche nach Beiträgen einer bestimmten Art zu vereinfachen. Bestimmte Kategorien werden weiters in Unterkategorien aufgeteilt, da diese ein zu weit gefächertes Genre an Anfragen umfassen.

Es existieren folgende Kategorien bzw. Unterkategorien:

- Mitteilung
 - Frage
 - Appell
 - Warnung
 - Empfehlung
 - Gefunden
- Suche
 - Hilfe
 - Verloren
- Ausleihen

- Event

Mitteilung: Die Kategorie der Mitteilung dient dazu die Nachbarn über ein bestimmtes Ereignis oder Meldung zu informieren oder zu befragen.

Suche: Die Kategorie der Suche dient dazu mit den Nachbarn im Falle einer Hilfesuche oder eines verloren gegangenen Objekts in Kontakt zu treten.

Ausleihen: Die Kategorie des Ausleihens dient dazu die Nachbarn nach der Erlaubnis, sich ein bestimmtes Werkzeug oder Objekt ausborgen zu dürfen, zu bitten.

Event: Die Kategorie des Events dient dazu die Nachbarn auf eine bestimmte Veranstaltung aufmerksam zu machen.

Tags

Als Tag wird ein Schlüsselwort beschrieben, was man an ein Informationsgut anhängen kann, um es besser beschreiben zu können und/oder besser auffindbar zu machen. In der App werden Tags als eine Erweiterung der Kategorien verwendet, um es Usern zu ermöglichen Ihren Beitrag einem selbstdefinierten Typ zuzuordnen.

Info

Jeder Beitrag hat eine eigene Sektion, wo wichtige Entscheidungsinformationen angegeben werden, wie der Stadtteil und die ungefähre Entfernung zum gegebenen Nachbarn und das Erstelldatum des Beitrags.

Kommentare

Die Kommentarfunktion ermöglicht es den Usern unter einem Beitrag Ihre Meinung, Feedback oder sonstiges zu hinterlassen.

Beitrag oder Kommentar Melden

Um auf unangebrachte Beiträge oder Kommentare schnell reagieren zu können, gibt es die Möglichkeit Beiträge oder Kommentare zu melden. Diese Meldungen werden auf Firestore gespeichert und können dann im Einzelnen überprüft werden. Fürs Melden muss ein Grund ausgewählt und eine genauere Beschreibung angegeben werden. Gründe fürs Melden eines Beitrags oder Kommentars:

- Unangebrachter Inhalt
- Belästigung
- Betrug
- Spam
- Sonstiges

4.2.6 Filter

Filtern nach Kategorien Ordnen nach Likes oder Datum Absteigend oder aufsteigend sortieren

4.2.7 Suche

4.2.8 Typesense

4.2.9 Algolia

diagram

Firestore Sync

Algolia SDK

4.2.10 Chat

package genommen warum

Flyer Package

4.2.11 Profil

Was wird angezeigt? Name, User Public Info, Posts von User

Profil Melden

4.2.12 Benachrichtigungen

Grund und Funktionsweise von Benachrichtigungen diagram beschreibung

4.2.13 Einstellungen

Email/Passwort ändern, Konto löschen, Sprache einstellen, Benachrichtigungen ein/aus-schalten

4.2.14 Feedback

feedback feature beschreiben

4.3 UI/UX Design

Das UI/UX-Design einer App ist von entscheidender Bedeutung für den Erfolg der App und die Zufriedenheit der Benutzer. Insbesondere bei einer Social-Media-Nachbarschafts-App ist ein gut durchdachtes UI/UX-Design unerlässlich, um eine positive Benutzererfahrung zu gewährleisten.

Ein gutes UI-Design ist wichtig, um sicherzustellen, dass die Benutzer die App einfach und intuitiv bedienen können. Es sollte eine klare Struktur und Navigation haben, damit die Benutzer schnell zu den gewünschten Funktionen gelangen können. Wenn die Benutzer die App als kompliziert oder verwirrend empfinden, werden sie möglicherweise frustriert und geben die Nutzung der App auf.

Auch ein gutes UX-Design ist wichtig, um sicherzustellen, dass die Benutzer mit der App zufrieden sind. Es sollte eine ansprechende und ansprechende Benutzeroberfläche bieten, die dem Benutzer ein angenehmes Nutzungserlebnis vermittelt. Wenn die Benutzer die App als langweilig oder uninteressant empfinden, werden sie möglicherweise nicht wiederkommen.

Darüber hinaus sollte das UI/UX-Design einer Social-Media-Nachbarschafts-App bestimmte Funktionen und Merkmale berücksichtigen, die für eine erfolgreiche Community-Plattform erforderlich sind. Beispielsweise sollte es einfach sein, Beiträge zu erstellen und zu teilen, auf Kommentare zu antworten und Nachrichten an andere Benutzer zu senden. Es sollte auch Möglichkeiten geben, um Benutzerprofile zu erstellen und zu verwalten sowie umfassende Datenschutz- und Sicherheitsfunktionen zu bieten.

Insgesamt ist das UI/UX-Design einer Social-Media-Nachbarschafts-App von entscheidender Bedeutung für den Erfolg der App und die Zufriedenheit der Benutzer. Es ist wichtig, dass das Design auf die Bedürfnisse und Anforderungen der Benutzer

zugeschnitten ist und ein einfaches, intuitives und ansprechendes Nutzungserlebnis bietet.

4.3.1 Inspiration

Während des Designprozesses für die App habe ich umfangreiche Recherchen im Bereich App-Design durchgeführt. Mein Ziel war es, die App so intuitiv wie möglich zu gestalten, um eine benutzerfreundliche Erfahrung zu gewährleisten. Dazu habe ich mich an bekannten Social-Media-Apps orientiert, wie zum Beispiel Twitter, Instagram und TikTok, die bereits auf dem Markt sehr erfolgreich sind und von vielen Menschen vertraut genutzt werden.

Darüber hinaus hat auch die erfolgreichste Nachbarschafts-App in Deutschland, "Nebenan", viel zur Grundlage der App beigetragen. Allerdings habe ich festgestellt, dass ihre App sehr kompliziert aufgebaut und unübersichtlich ist, was für uns eine Chance darstellte, es besser zu machen.

Zusätzlich haben wir uns von Websites wie Dribbble und Mobbin inspirieren lassen, die uns geholfen haben, ein einfaches und schlichtes Design für die App zu entwickeln. Insgesamt war die Recherche und Inspiration für das Design der App ein wichtiger Schritt, um sicherzustellen, dass unsere Benutzer eine ansprechende und intuitive Erfahrung haben.

4.3.2 Prototyping

Prototyping ist ein wichtiger Schritt bei der Entwicklung von mobilen Apps. Es ermöglicht Entwicklern, Designern und anderen Stakeholdern, eine frühzeitige Vorstellung davon zu bekommen, wie die App funktionieren wird und wie sie aussehen wird. Durch die Erstellung eines Prototyps können auch Fehler im Design und in der Funktionalität identifiziert werden, bevor die App in die eigentliche Entwicklung geht.

Es gibt verschiedene Arten von Prototypen, darunter Low-Fidelity- und High-Fidelity-Prototypen. Low-Fidelity-Prototypen sind einfache Skizzen oder Wireframes, die nur die grundlegenden Funktionen und das Layout der App abbilden. High-Fidelity-Prototypen sind detaillierter und können interaktive Funktionen und Designelemente enthalten.

Es gibt viele Tools und Plattformen, die zur Erstellung von Prototypen verwendet werden können. Ein Beispiel ist Adobe XD, das es Entwicklern und Designern ermöglicht,

Framer ist eine Software, die es Benutzern ermöglicht, schnell und einfach ansprechende Prototypen von mobilen Anwendungen und Websites zu erstellen. Das Tool wurde ursprünglich als Prototyping-Tool für Designer entwickelt, um Designs schnell zu testen und zu verfeinern, bevor sie in die Entwicklung übergehen.

Erfolgreiche Apps wie Spotify haben Framer im Rahmen ihres Prototyping-Prozesses verwendet, um schnell und effizient funktionierende App-Designs zu erstellen. Framer ist eine schnelle und effiziente Möglichkeit, um Ideen in die Tat umzusetzen, ohne sich durch langwierige Entwicklungsprozesse zu quälen.

Framer wurde während des Hackerthon Linz hackt verwendet, um den ersten Prototypen zu erstellen. Die Wahl von Framer war aufgrund seiner Geschwindigkeit und Effizienz in der Erstellung von funktionsfähigen App-Designs und der Erfahrung, die der Benutzer bereits mit dem Tool hatte, getroffen worden.

Seit der Erstellung des ersten Prototyps hat sich das Geschäftsmodell von Framer jedoch geändert. Es ist jetzt ein Website-Baukasten, der es Benutzern ermöglicht, einfach und schnell ansprechende Websites zu erstellen, ohne Kenntnisse in der Webentwicklung zu benötigen. Obwohl es nun als Website-Baukasten fungiert, behält Framer immer noch einige seiner Kernfunktionen als Prototyping-Tool bei, was es zu einer guten Option für Designer und Entwickler macht, die schnell Prototypen erstellen möchten.

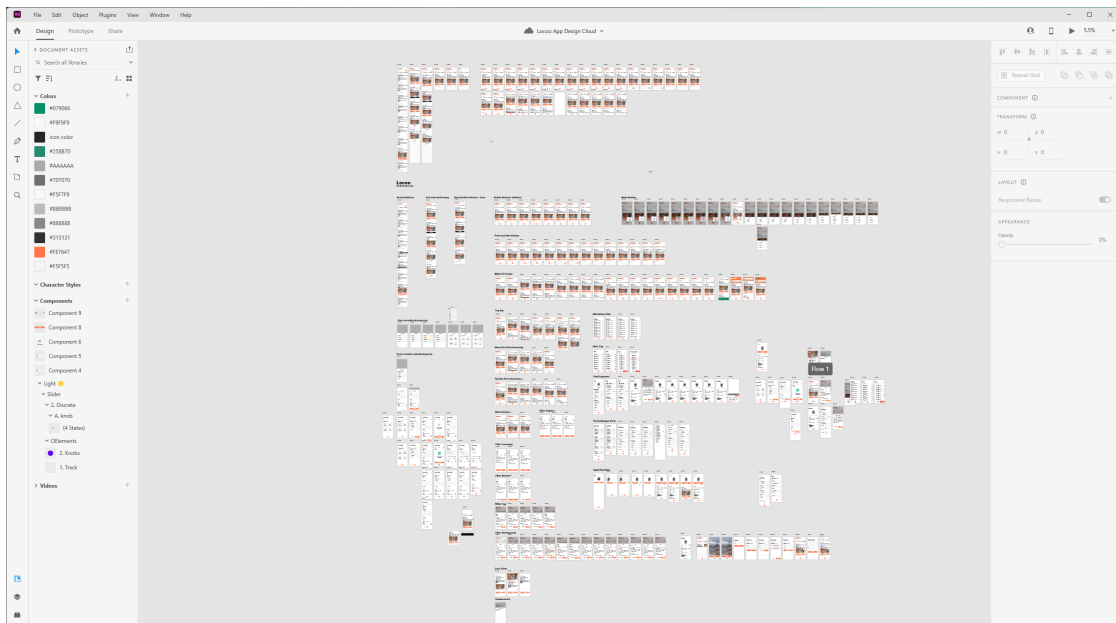
Insgesamt hat Framer gezeigt, dass es ein schnelles und effektives Tool ist, um Designideen in die Tat umzusetzen. Obwohl es nun als Website-Baukasten fungiert, ist es immer noch eine nützliche Option für Designer und Entwickler, die schnell und einfach Prototypen erstellen möchten.

Adobe XD

Als ich zum ersten Mal mit Framer arbeitete, war ich begeistert von den vielen Features und Möglichkeiten, die es bietet. Aber im Laufe der Zeit stellte ich fest, dass es für mein Projekt zu aufwändig war und ich nach einer einfacheren Lösung suchte. So bin ich auf Adobe XD umgestiegen, ein Tool, mit dem ich schon jahrelange Erfahrung hatte.

Obwohl Adobe XD nicht so viele Features wie Framer bietet, ist es aufgrund seiner Einfachheit und Benutzerfreundlichkeit ein ideales Tool, um schnell einen Prototypen zu erstellen. Ich hatte alle Hauptscreens fertig gestaltet, bevor wir mit der Entwicklung mit Flutter begannen. Die anderen wichtigen Screens habe ich dann später gestaltet,

als ich mit Flutter vertrauter war und besser abschätzen konnte, wie aufwändig es in Flutter umzusetzen war.



Wie man auf dem obigen Bild sehen kann, ist unser Prototyp eine Sammlung verschiedener Designs und Evolutionen des Designs. Adobe XD war für unser Projekt perfekt, da ich der alleinige Designer war und somit keine Kollaborationsfunktion benötigte. Allerdings, da ich jetzt mit dem bekanntesten Designtool Figma vertraut bin und es billiger ist, wenn mehrere Personen an einem Design arbeiten und auch mehr Funktionen bietet, würde ich aktuell mit Figma weiterarbeiten.

Zusammenfassend lässt sich sagen, dass Adobe XD ein mächtiges Tool für UI/UX-Designer ist, das einfach zu bedienen und ideal für kleine bis mittelgroße Projekte ist. Es bietet zwar nicht so viele Funktionen wie andere Tools, ist aber für schnelle Prototypenerstellung und einfache Zusammenarbeit mit Entwicklern und Stakeholdern sehr gut geeignet.

4.3.3 Design

allg.

Farben

farben history warum orange

Icons

foto von den benutzen icons welche icons warum

Fonts

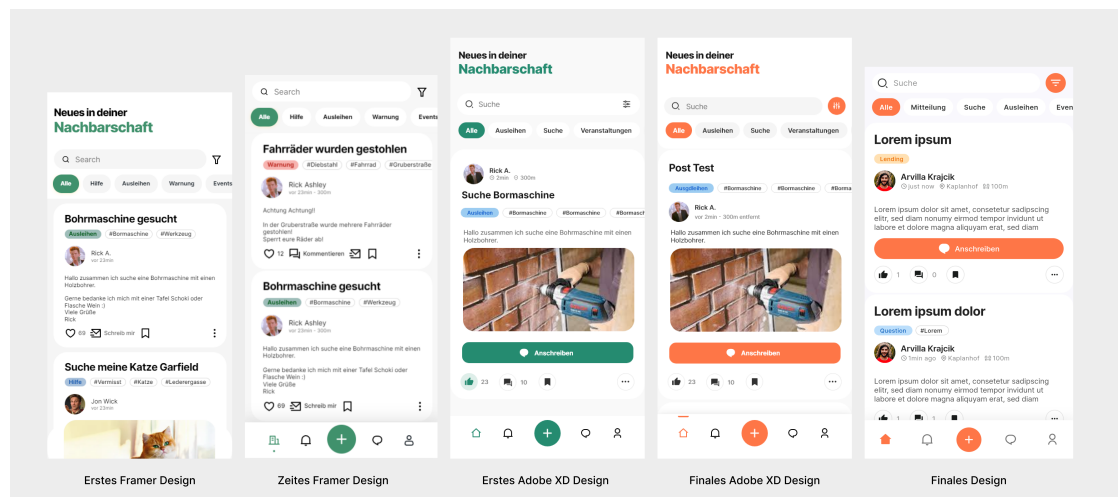
fotos von fonts welche fonts wie ist die typographie aufgebaut

Logo

logo foto anforderungen design ideen logo history

4.3.4 App Design

Design Historie



ein-

gehen mit fotos und beschreibung auf die haupt screens der app

4.3.5 Websites Design

2-3 nochba.at design foto beschreibung

4.4 Backend

4.4.1 Cloud

4.4.2 Firebase

Firebase SDKs

Firebase-Tools

Firebase Authentication

Email und Passwort authentication

Cloud Firestore

NoSQL-Database, Dokument-basierte Speicherung, Subcollections

Cloud Storage

Speicherung von Profilbilder und Post-Bilder

Firebase Cloud Functions

jeder schreibt über seine CF

4.4.3 Algolia Search

4.4.4 Algolia SDK

Typesense Search

4.4.5 Typesense SDK

5 Evaluating getroffener Entscheidungen

6 Zusammenfassung

Aufzählungen:

- Itemize Level 1
 - Itemize Level 2
 - Itemize Level 3 (vermeiden)
- 1. Enumerate Level 1
 - a. Enumerate Level 2
 - i. Enumerate Level 3 (vermeiden)

Desc Level 1

Desc Level 2 (vermeiden)

Desc Level 3 (vermeiden)

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

| | | |
|---|-----------------------------------|---|
| 1 | Security Rules Beispiel | 7 |
|---|-----------------------------------|---|

Anhang