

Nochba – App für Nachbarschaftshilfe

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Informatik

Eingereicht von:

Arsham Edalatkhan
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
1.1 Motivation	1
1.2 Aufgabenstellung	1
1.3 Zusammenfassung des Projektergebnisse	1
1.4 Projektvorgehensmodell	2
2 Technologien	4
2.1 Technologieevaluierung	4
2.2 Technologieentscheidung	9
3 Systemarchitektur	11
3.1 Flutter	12
3.2 Firebase	12
3.3 Algolia Search	16
4 Umsetzung	19
4.1 Continuous Integration/Delivery	19
4.2 Mobile Anwendung	22
4.3 UI/UX Design	29
4.4 Website	44
4.5 Backend	46
4.6 Email	54
5 Evaluating getroffener Entscheidungen	56
6 Zusammenfassung	57
Literaturverzeichnis	VI
Abbildungsverzeichnis	VII

Tabellenverzeichnis **VIII**

Quellcodeverzeichnis **IX**

Anhang **X**

1 Einleitung

1.1 Motivation

1.2 Aufgabenstellung

1.3 Zusammenfassung des Projektergebnisse

In den letzten Monaten haben wir an einer Reihe von Wettbewerben teilgenommen oder gewonnen, die unsere Kompetenz im Bereich der nachhaltigen Innovation und sozialen Unternehmerschaft unterstreichen. Hier sind die Details:

Linz-hACKT (11. bis 13. März 2023):

Dies war ein Hackathon, der sich mit der Schaffung von Ideen und Visionen für eine klimaneutrale Stadt Linz bis 2040 beschäftigte. Wir haben an diesem Wettbewerb teilgenommen und konnten den ersten Platz erreichen. Dies hat uns die Chance gegeben, gute Verbindungen mit der Stadt Linz, dem Innovationshauptplatz Linz und Open Common Linz aufzubauen. Außerdem hatten wir die Chance, am 30. Jänner 2023 eine Präsentation für den Linzer Bürgermeister Klaus Luger und die Magistratsdirektorin Ulrike Huemer zu halten.

mPreneur Austria (22. September 2022):

Im Rahmen des Projekts „mPreneur“ arbeiteten wir zusammen mit acht Organisationen weltweit, um ICT-Kompetenzen bei jungen Menschen aufzubauen. Wir haben an diesem Projekt teilgenommen und den ersten Platz gewonnen.

mPreneur Social Mobile Entrepreneurship (04. bis 09. November 2022):

Dieses Projekt förderte die Kapazitätsentwicklung junger angehender Unternehmer und Jugend-CSOs. Ich, Arsham Edalatkahah, hatte die Chance, die Nochba App während des interkontinentalen Wettbewerbs "Social Mobile mPreneur" in Nord Mazedonien zu pitchen. Ich absolvierte im Vorfeld zwischen 6 und 8 Meetings mit Präsentationsexper-

ten und konnte dadurch wertvolle Erfahrungen im Bereich der Präsentationstechnik sammeln.

1.4 Projektvorgehensmodell

1.4.1 Projektorganisation

Discord

Die primäre Kommunikationsplattform zwischen dem Team und den Diplomarbeit-betreuer ist Discord. Da die Arbeit in den Sommerferien 2022 begonnen hat und zu diesem Zeitpunkt Corona-Regelungen galten, musste eine Meeting-Plattform gefunden werden, die sowohl Video- als auch Text-Chat ermöglicht. Aufgrund der Vertrautheit mit Discord und der Tatsache, dass es kostenlos ist, fiel die Entscheidung leicht.

Während der Diplomarbeit wurden über 40 Text-Kanäle erstellt, um alle relevanten Informationen, Notizen und Dokumentation zu speichern. Die Meeting-Notizen wurden jedoch auf ClickUp gespeichert.

ClickUp

Um das Projekt effizient zu organisieren, wird nicht nur eine WhatsApp-Gruppe genutzt, sondern auch das Projektmanagement-Tool ClickUp. Die Entscheidung für ClickUp fiel leicht, da bereits viele Projektmanagement-Programme ausprobiert wurden und ClickUp bei den letzten Projekten am besten abgeschnitten hat. Außerdem ist es sehr intuitiv und einfach zu erlernen, was von beiden Team bestätigt wird.

In der Diplomarbeit wurden viele Mentoring-Sessions, Meetings mit dem Diplomar-beitbetreuer und weitere Termine wie Wettbewerbe vereinbart. Ohne eine geeignete Methode zur Verwaltung all dieser Termine kann es schnell unübersichtlich werden. Deshalb wurde das Kalender-Feature von ClickUp genutzt, um alle Einträge abzubilden, wie in Abbildung 1 zu sehen ist. Der Kalender wurde automatisch mit den persönlichen Kalendern synchronisiert, wie beispielsweise einem Google Kalender. Dadurch wurden nie Termine verpasst, was insbesondere dem Projektleiter sehr geholfen hat, da er nicht jeden daran erinnern musste, alle Termine in seinem Kalender einzutragen.

Ein weiterer wichtiger Punkt war das Task-Management in dem Projekt. Da im Projekt schnell viele Aufgaben koordiniert werden mussten, war es sinnvoll, alle Aufgaben in

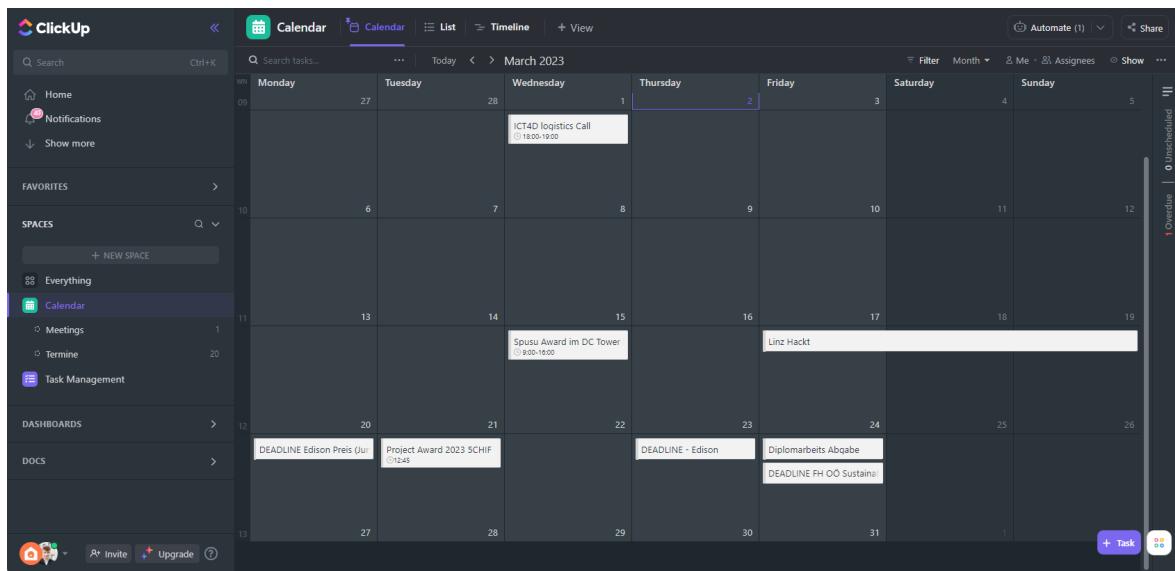


Abbildung 1: ClickUp Dashboard Kalender

einer simplen To-Do-Liste abzubilden. Vor den Semesterferien wurde beispielsweise eine eigene Abteilung für Ferien-Tasks erstellt, um einen Überblick über alle Aufgaben während der Semesterferien zu haben. Obwohl die Tasks-To-Listen nicht aktiv genutzt wurden, war es sehr praktisch, um den Fortschritt zu überwachen und sicherzustellen, dass alles rechtzeitig erledigt wurde.

Zusammenfassend war ClickUp eine große Bereicherung für das Projektmanagement, da es viel Zeit bei der Verwaltung von Terminen und Aufgaben gespart hat. Auch die Benutzung als neuer Benutzer war sehr einfach, sodass keine lange Einarbeitungszeit nötig war. Besonders faszinierend ist, dass ClickUp sehr intuitiv ist und viele nützliche Features bietet.

2 Technologien

2.1 Technologieevaluierung

Eine grundlegende Komponente des Projektbeginns besteht in der Evaluierung der Technologie. Hierbei wurde bewusst ein angemessener Zeitrahmen zugrunde gelegt, um eine reibungslose und effiziente Entwicklungsphase zu gewährleisten und eventuelle Unzulänglichkeiten im späteren Verlauf des Projekts zu vermeiden. Die definierten Anforderungen umfassen dabei folgende Aspekte:

- Entwicklung von Apps für Android und iOS
- Möglichkeit zur zukünftigen Veröffentlichung der Anwendung als Webapp
- Unterstützung von Datenbankabfragen auf Grundlage von Geo-Koordinaten
- Integrierte Authentifizierungs- und Autorisierungsschicht
- Realtime-Datenbank, um Echtzeit-Chatrooms für Konversationen zu führen.

2.1.1 Frontend

Im Bereich der Frontend-App-Entwicklung stehen verschiedene Möglichkeiten zur Verfügung, um eine App zu programmieren:

- Native Entwicklung für jede Plattform (iOS, Android, Web)
- Nutzung eines Frameworks für plattformübergreifende Entwicklung von Apps

Die native Entwicklung für verschiedene Plattformen wie iOS, Android und Web ermöglicht die spezifische Programmierung einer App in der nativen Sprache der Plattform. Dabei werden für iOS-Apps die Programmiersprache Swift und für Android-Apps Java oder Kotlin verwendet, während für Web-Apps HTML, CSS und JavaScript zum Einsatz kommen.

Durch die native Entwicklung kann eine optimale Performance der App erreicht werden, da sie die native Funktionalität des Geräts vollständig nutzen kann. Entwickler haben die Möglichkeit, auf alle nativen APIs zuzugreifen und die Benutzeroberfläche der App an das

jeweilige Betriebssystem anzupassen. Dadurch kann eine höhere Benutzerfreundlichkeit und eine bessere Integration in das Betriebssystem erreicht werden.

Allerdings erfordert die native Entwicklung spezialisierte Kenntnisse in den verschiedenen Plattformen und Programmiersprachen. Mehrere Entwickler mit unterschiedlichen Fachkenntnissen sind möglicherweise erforderlich, um die App für verschiedene Plattformen zu entwickeln. Dadurch kann die native Entwicklung teurer und zeitaufwändiger sein als andere Entwicklungsansätze.

Ein Framework für plattformübergreifende Entwicklung ermöglicht die Erstellung einer einzigen Codebasis, die auf verschiedenen Plattformen wie iOS, Android und dem Web ausgeführt werden kann. Ein solches Framework nutzt häufig eine gemeinsame Programmiersprache wie JavaScript und bietet eine Vielzahl von Bibliotheken und Tools, um die Entwicklung zu erleichtern. Das Framework ermöglicht es Entwicklern, schnell und effizient Apps zu erstellen, ohne sich mit den Unterschieden in den Plattformen und Programmiersprachen auseinandersetzen zu müssen. Ein weiterer Vorteil von Frameworks für plattformübergreifende Entwicklung besteht darin, dass sie eine schnellere Markteinführung ermöglichen und den Entwicklungsaufwand reduzieren können. Allerdings kann die Verwendung eines Frameworks die Performance beeinträchtigen, da es schwierig sein kann, die App für jede Plattform zu optimieren.

Infolge einer schnellen Entscheidung haben wir beschlossen, dass die Bewältigung dieser Angelegenheit mithilfe nativer Programmierung nicht erfolgen wird. Die Unvertrautheit eines jeden Mitglieds unseres Teams mit den Plattformen würde einen doppelten Arbeitsaufwand bedeuten, der für ein kleines Team von nur drei Personen zeitlich unverhältnismäßig wäre.

Um eine effektive und plattformübergreifende Entwicklung zu gewährleisten, haben wir uns dazu entschieden, ein Cross-Platform-Framework zu verwenden. Wir haben die Statistik in der Abbildung 2 betrachtet, welche Plattformen am meisten genutzt werden und dabei festgestellt, dass Flutter mit 42 Prozent das meistgenutzte Framework ist und die Tendenz steigend ist. Auf Platz 2 liegt React Native, das fast genauso beliebt ist.

Im Rahmen der vorliegenden Untersuchung wurden die wichtigsten Informationen zu verschiedenen Frameworks in einer Vergleichstabelle zusammengefasst. Es wurde dabei festgestellt, dass sämtliche Plattformen in der Lage sind, die gestellten Anforderungen zu erfüllen.

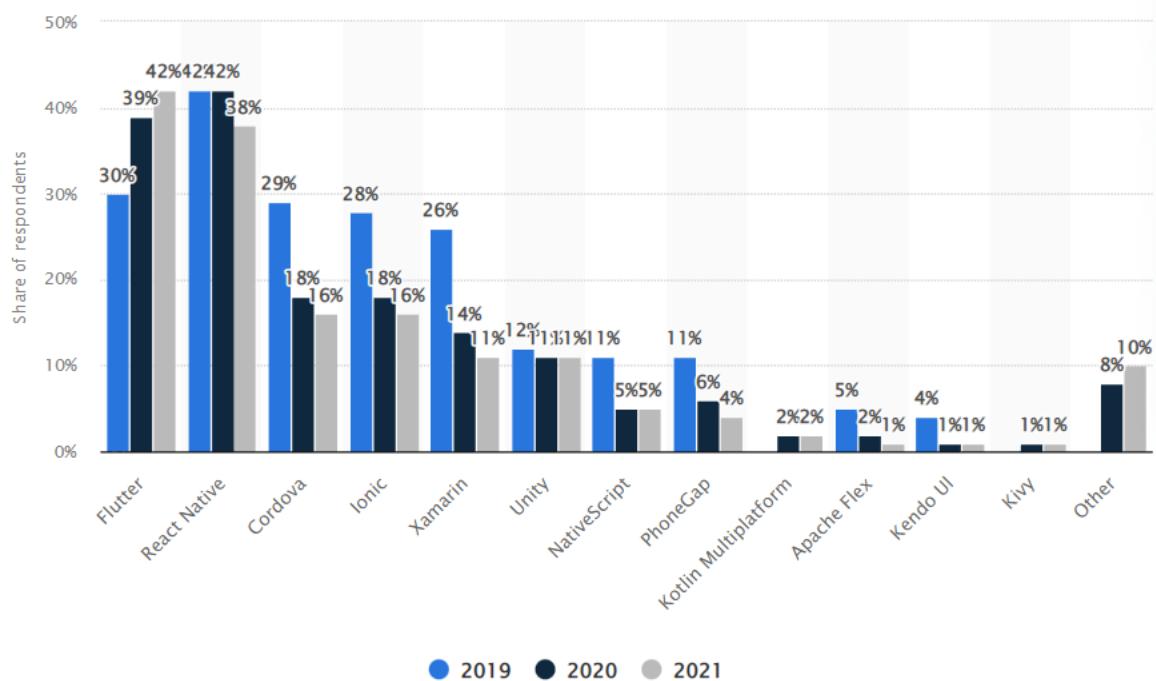


Abbildung 2: Statistiken zur plattformübergreifenden Nutzung von Statista [1]

Insbesondere hat das Framework Xamarin aufgrund der jahrelangen Programmiererfahrung des Teams mit der Sprache C# ein gewisses Interesse geweckt. Es ist jedoch zu erwähnen, dass auch das Framework Dart aufgrund der ähnlichen Syntax zu C# eine geeignete Option für das Team darstellt.

Programmiersprachen React Native ist zwar in JavaScript geschrieben, aber man kann keine React-Komponenten importieren und verwenden. Wenn man jedoch bereits Erfahrung mit React.js hat, kann man sich als Entwickler leichter in React Native einarbeiten. Die letzten beiden Plattformen Cordova und Ionic basieren auf HTML, CSS und JavaScript, was den Vorteil bietet, dass man auf jede JavaScript-Library zugreifen kann.

UI Widgets Flutter hat als UI-Widget-Library sowohl Material Design 2 und 3 als auch eine iOS Library, die das native iOS-Design 1:1 abbildet. Dadurch kann der Nutzer den Unterschied zwischen nativen und Flutter-Designs kaum erkennen. Xamarin hat ebenfalls eine UI-Library namens Xamarin.Forms, die auf allen Plattformen funktioniert und native UI-Libraries für jede Plattform enthält.

Stand 3.2022	Flutter	React Native	Cordova	Ionic	Xamarin
Version	2.10	0.68	11.0.0	6.0.10	5.10
iOS	iOS 8+	iOS 9+	iOS 8+	iOS 8+	iOS 8+
Android	Android 4.1	Android 4.1+	Android 4.1+	Android 4.1+	Android 4.1+
Web	✓	✓	✓	✓	✓
Language Support	Dart	JavaScript	HTML, CSS, JavaScript	HTML, CSS, JavaScript	C#
UI Widgets	Material Design, Cupertino (iOS)	Jede	-	Material Design	Xamarin Forms, Native UI
Hot Reload	Yes	Yes	No	Yes	Yes
License	BSD 3-Clause	MIT	Apache 2.0	MIT	MIT
Package Manager	Pub	NPM, Yarn	NPM	NPM	NuGet
Packages verfügbar 2.2023	32914	1216	6880	400+	1588
Github Stars 2.2023	150k	108k	5k	48k	15k

Abbildung 3: Vergleich der mobilen Frameworks nach Stand 3.2022 [2, 3, 4, 5, 6]

Hot Reload Ein wichtiger Faktor bei der Wahl des Frameworks ist auch der Hot Reload. Cordova bietet diese Funktion nicht, alle anderen Plattformen hingegen schon, was für uns ein dealbreaker ist.

Package Manager In der vorliegenden Vergleich wurden verschiedene Package-Manager miteinander verglichen. Nach sorgfältiger Analyse hat sich gezeigt, dass der Package-Manager von Flutter am überzeugendsten ist. Ein ausschlaggebender Faktor hierfür war, dass jeder einzelne Package eine Beispielanwendung beinhaltet. Zudem bietet der Package-Manager die Möglichkeit, durch die Verwendung von Pub Points die Leistung und Qualität des Packages anhand verschiedener Kriterien zu bewerten. Ein weiterer Vorteil besteht darin, dass man leicht erkennen kann, welche Plattformen das jeweilige Package unterstützt. Zusammenfassend lässt sich sagen, dass der Package-Manager von Flutter aufgrund seiner umfangreichen Funktionen und Tools eine herausragende Wahl darstellt.

Obwohl npm der meistgenutzte Package-Manager ist, ist er nicht spezifisch auf ein bestimmtes Framework ausgerichtet und es fehlen einige Funktionen, die Pub hat. In

der C#-Entwicklung ist NuGet bereits bekannt, jedoch wirkt er veraltet und weist ebenfalls einige Funktionslücken im Vergleich zu Pub auf.

Packages Ein weiterer wichtiger Faktor bei der Wahl des Frameworks sind die verfügbaren Packages. Flutter bietet hier eine große Auswahl, wobei zu beachten ist, dass bei Frameworks, die auf JavaScript basieren, auf alle JavaScript-Libraries zugegriffen werden kann. Insgesamt sind die meisten Flutter-Packages auf mobile Apps ausgerichtet.

Github Stars Die Community wurde anhand der Anzahl der Github-Stars gemessen, wobei Flutter mit gut einem Drittel vor React Native liegt. Obwohl Flutter noch nicht so lange auf dem Markt ist, spricht dies dafür, dass Entwickler mit Flutter sehr zufrieden sind.

Performance

Die Performance ist ein wichtiger Faktor bei der Nutzung von Cross-Platform-Frameworks, da sie aufgrund ihrer Nicht-Nativität variieren kann. In einem Forschungspapier [7] wurden alle genannten Plattformen bis auf Cordova verglichen, wobei ein aussagekräftiger Vergleich in Punkt 2.6 durchgeführt wurde. Die Ergebnisse zeigen, dass Flutter in Bezug auf die Performance den ersten Platz belegt, gefolgt von React Native auf Platz 2. Im Gegensatz dazu schneidet Xamarin in diesem Vergleich nicht so gut ab. Die letzte Position nimmt Iconic ein, was aufgrund der Tatsache, dass es lediglich eine HTML-, CSS- und JavaScript-Seite in einem Webview anzeigt, nachvollziehbar ist. Im Vergleich zu den anderen Plattformen kann es aufgrund des Fehlens eigener Renderer nicht mithalten. Wichtig ist zu beachten, dass die Performance-Tests im Dezember 2021 durchgeführt wurden und seitdem hat sich die Geschwindigkeit des Frameworks bis zum aktuellen Zeitpunkt erheblich verbessert.

2.1.2 Backend

Dank Martin Hausleitners fundiertem Wissen in verschiedenen Backend-Technologien fiel uns die Entscheidung für unser Projekt leichter. Bei der Auswahl konzentrierten wir uns auf "Backend-as-a-Service"-Plattformen, da sie bereits zahlreiche Funktionen implementiert haben und das Team somit nicht alles von Grund auf programmieren mussten.

Bei der Auswahl für das Projekt wurden drei Technologien berücksichtigt: Supabase, Firebase und Appwrite. Jede dieser Optionen stellt SDKs für Flutter und React Native bereit. Zudem verfügen alle über eine Datenbank, die Geoqueries unterstützt, die für das zentrale Merkmal des Radius der Beiträge benötigt wird. Zusätzlich bietet jede Plattform eine Speicherlösung für Beitragsfotos und andere Funktionen.

Supabase hat zum Zeitpunkt der Evaluierung im Februar 2022 jedoch noch keine Cloud-Functions wie Firebase und Appwrite, was es schwierig machte, 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 blieben Appwrite und Firebase als die beiden besten Optionen übrig. Obwohl Appwrite nicht mit den Features von Firebase mithalten konnte, ist Appwrite 100 Prozent Open Source und Firebase Closed Source, was für Appwrite spricht. Da wir uns bei unserem Cross-Platform-Framework noch nicht sicher waren, war es uns nun leichter, uns für Firebase zu entscheiden, da es das beste Flutter-SDK hatte, was man anhand der verfügbaren Features erkennen konnte.

2.2 Technologieentscheidung

Die Kombination von Flutter und Firebase harmonieren gut. Flutter ist schnell und hat eine breite Palette von Erweiterungen für mobile Anwendungen sowie eine starke Community. Flutter wird voraussichtlich Marktführer, was ein gutes Gefühl bezüglich der Entscheidung gibt. Zu Beginn gab es Zweifel bei der Arbeit mit Flutter aufgrund der vielen Klammern, die schnell unübersichtlich wurden. Jedoch konnte die Herausforderung dank der super VS Code-Integration mit einem Reformat-Feature gemeistert werden. Auch nach einem Jahr Entwicklung mit Flutter ist es immer noch beeindruckend, wie einfach Features zu implementieren sind und wie straight-forward die Entwicklung ist. Der Syntax von Dart, der Programmiersprache hinter Flutter, ist schnell erlernbar und ermöglicht es, schnell viele Programmierpatterns zu beherrschen.

Firebase hat sich als effektive Wahl erwiesen aufgrund der schnellen und einfachen Entwicklungsmöglichkeiten. Die Skalierung wird automatisch gehandhabt, was ein großer Vorteil ist. Firebase bietet nach wie vor viele erstklassige Features. Allerdings gibt es zwei Nachteile, die die Entscheidung für das Backend beeinflussen könnten. Die Firestore-Datenbank hat keine integrierte Suchfunktion, weshalb Algolia verwendet wurde. Algolia bietet fortgeschrittene Suchfunktionen, die andere Datenbank-Suchmaschinen nicht

haben, aber es ist keine perfekte Lösung. Transparenz und Open Source sind wichtige Themen in der Diplomarbeit. Firebase ist allerdings Closed Source, was die bei der Transparenz einschränkt. Aus diesem Grund würde aktuell Supabase bevorzugt, da es Cloud Functions und eine Open-Source-Option bietet. Supabase verwendet PostgreSQL, das bereits eine Suchfunktion integriert hat.

3 Systemarchitektur

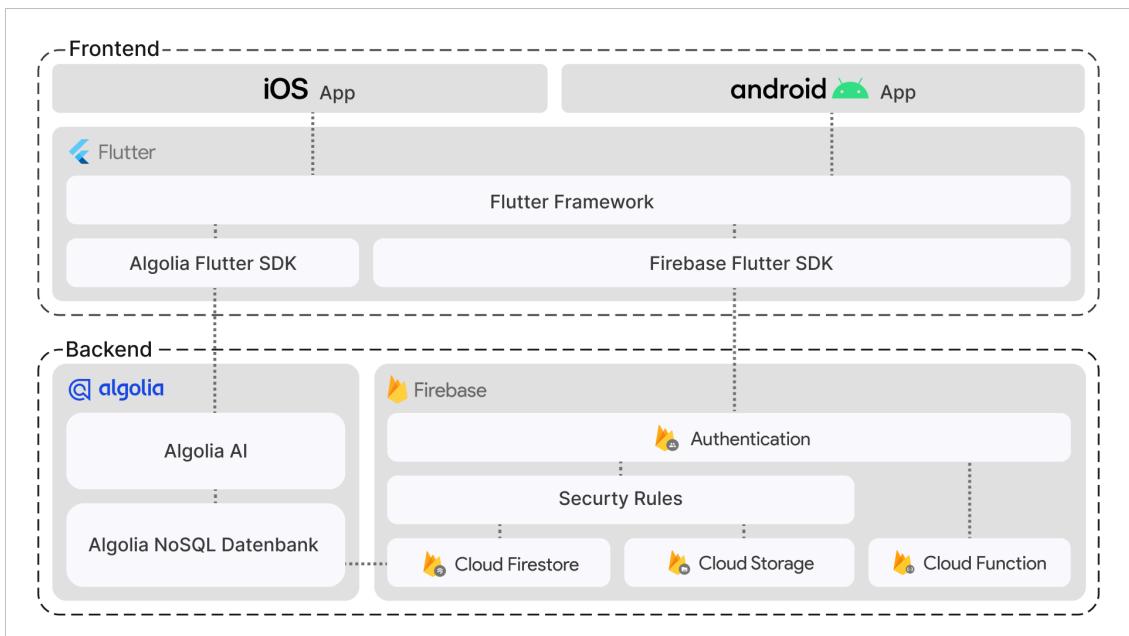


Abbildung 4: Systemarchitektur Diagramm

Die vorliegende Abbildung 4 veranschaulicht vereinfacht unsere Systemarchitektur. Der obere Teil stellt das Frontend dar, welches vollständig im Flutter Framework unter Verwendung der Programmiersprache Dart entwickelt wurde. Zur Kommunikation mit dem Backend wurden die Algolia Flutter SDK und Firebase Flutter SDK genutzt. Zur Vereinfachung wurden die Funktionen der Firebase Flutter SDK in einem einzigen Paket zusammengefasst, während Firebase für jedes Feature ein eigenes Package bereitstellt. Die Flutter-App wird anschließend für iOS und Android kompiliert.

Im Backend wurde besonderer Wert auf die Implementierung von Sicherheitsmaßnahmen gelegt. Der erste Sicherheitslayer ist die Firebase-Authentifizierung, die das gesamte Authentifizierungssystem für den Zugriff auf das Backend regelt. Firebase Cloud Functions können nur von einem registrierten Account ausgeführt werden. Besondere Cloud Functions verlangen darüber hinaus eine zusätzliche Verifizierung der Adresse des Nutzers.

Wir nutzen Firestore, eine NoSQL-Datenbank von Firebase, um unsere Daten zu speichern. Um auf die Firestore-Datenbank und den Cloud Storage zugreifen zu können,

müssen zuerst die Firestore Security Rules durchlaufen werden, die sämtliche Zugriffsrechte auf die Dokumente in der Datenbank oder dem Cloud Storage regeln. Mehr dazu findet sich unter Security Rules.

Firestore verfügt nicht über eine integrierte Funktion zur vollständigen Durchsuchung aller Dokumente, weshalb für die Suche nach Inhalten eine externe Suchmaschine wie Algolia erforderlich ist. Durch die Verbindung von Cloud Firestore mit der Algolia SQL-Datenbank wird sichergestellt, dass die Datenbanken synchronisiert werden. Ein bedeutender Vorteil dieser Vorgehensweise, der möglicherweise zunächst als Nachteil betrachtet werden könnte, besteht darin, dass wir über zwei Datenbanken verfügen, die dieselben Daten enthalten. Die Algolia-Datenbank verfügt jedoch über eine integrierte Suchfunktion, die die Suchgeschwindigkeit und -qualität verbessert, indem nur die Beitrag-Titel, -Beschreibungen und -Tags gespiegelt werden, anstatt die gesamte Datenbank zu replizieren. Weitere Informationen zu Algolia Search ist in dem Abschnitt 3.3 zu finden.

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[8] 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[9][10][11], 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 diente.

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[12] 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

Algolia ist eine Suchtechnologie, die es Entwicklern ermöglicht, relevante und schnelle Suchergebnisse für ihre Anwendungen bereitzustellen. Diese Technologie wurde speziell für moderne Anforderungen im Bereich der Suche entwickelt und bietet Funktionen wie automatisches Vervollständigen, Sortierung nach Relevanz und Suche nach Kategorien.

Die Integration von Algolia in eine App-Architektur ist einfach durchzuführen und benötigt keine langen Zeiträume. Es stellt eine RESTful API zur Verfügung, die direkt von der Anwendung aufgerufen werden kann. Es gibt auch integrierte Lösungen für verschiedene Backend-Systeme wie Firebase, was die Integration noch einfacher macht.

Durch die Verwendung von Algolia kann eine Anwendung schnelle und relevante Suchergebnisse bereitstellen, was das Nutzererlebnis verbessert. Es gibt auch eine Vielzahl von Tools und Funktionen, die Entwicklern bei der Optimierung ihrer Suchergebnisse helfen, um die besten Ergebnisse für ihre Nutzer zu erzielen.

Ein Firebase Algolia-Extension ist eine vorgefertigte Integration zwischen Firebase, einer Backend-as-a-Service-Plattform, und Algolia, einer Search-as-a-Service-Plattform. Durch diese Integration können Entwickler leistungsstarke Suchfunktionen in ihre Firebase-Anwendungen integrieren.

Firebase bietet eine Reihe von Tools und Diensten zur Entwicklung und Hosting von mobilen und Webanwendungen, darunter Authentifizierung, Echtzeitdatenbank, Cloud-Speicherung, Hosting und mehr. Algolia hingegen bietet eine leistungsstarke Suchmaschine, die in Echtzeit nach großen Datenmengen suchen kann, sowie Funktionen wie Tippfehler-Toleranz, Autocomplete und Relevanzanpassung.

Als ich meine Diplomarbeit schrieb, habe ich mich für Algolia entschieden, weil es eine leistungsstarke und benutzerfreundliche Suchmaschine bietet, die einfach zu integrieren ist. Algolia ermöglicht es Entwicklern, schnell eine leistungsstarke Sucherfahrung für ihre Benutzer aufzubauen.

Der Firebase Algolia-Extension ermöglicht es Entwicklern, Daten automatisch aus der Firebase-Datenbank in den Algolia-Index zu synchronisieren, was es ihnen ermöglicht, schnell eine leistungsstarke Sucherfahrung für ihre Benutzer einzurichten. Entwickler können die Indizierungseinstellungen anpassen und das vorgefertigte Such-Widget in ihre

Anwendung integrieren. Die Erweiterung integriert auch Firebase-Sicherheitsregeln, um den Zugriff auf Suchergebnisse basierend auf Benutzerauthentifizierung und -autorisation zu steuern.

Algolia bietet verschiedene Preismodelle an, die verschiedenen Anwendungsfällen und Budgets entsprechen. Für kleinere Projekte gibt es einen kostenlosen Plan, der bis zu 10.000 Datensätze und 100.000 Operationen pro Monat bietet. Für größere Projekte bietet Algolia kostenpflichtige Pläne, die mehr Datensätze und Operationen pro Monat sowie zusätzliche Funktionen wie Analytics und Personalisierung bieten.

Kunden können auch zusätzliche Sucheinheiten erwerben, wenn sie die Grenzen ihres Plans überschreiten. Die Kosten für zusätzliche Sucheinheiten variieren je nach Plan und Anzahl der erworbenen Einheiten.

Zusammenfassend basiert die Preisgestaltung von Algolia auf einem nutzungsabhängigen Modell, und Kunden zahlen basierend auf der Anzahl der Suchanfragen und der Komplexität dieser Anfragen, gemessen in Sucheinheiten. Algolia bietet verschiedene Preismodelle an, und Kunden können zusätzliche Sucheinheiten erwerben, wenn sie die Grenzen ihres Plans überschreiten.

Quellen:

- Firebase: <https://firebase.google.com/>
- Algolia: <https://www.algolia.com/>
- Firebase Algolia-Erweiterung: <https://firebase.google.com/products/extensions/firestore-algolia-search>
- Algolia-Preisgestaltung: <https://www.algolia.com/pricing/>

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/>

Firebase Cloud Function

Beschreibung der Firebase 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

iOS Build

Unser Ziel bestand darin, eine iOS-App im App Store zu veröffentlichen und eine Android-App im Play Store anzubieten. Da Martin Hausleitner der einzige im Team war, der ein iPhone besaß und über Erfahrungen mit Apple verfügte, beschäftigte er sich mit dem Build-Prozess der iOS-App.

Zur Erstellung einer Flutter iOS-App ist ein Mac mit Xcode sowie ein Apple-Entwicklerkonto erforderlich. Da jedoch niemand im Team über ein solches Konto verfügte, gestaltete sich der Build-Prozess äußerst schwierig. Glücklicherweise konnte Martin Hausleitner einen alten iMac von seiner Familie ausleihen, auf dem er arbeiten konnte. Obwohl das Erstellen einer iOS-App auf den ersten Blick einfach erscheint, stellte es sich als eine der größten Herausforderungen in diesem Projekt heraus.

Da MacOS und Xcode dem Entwickler nicht vertraut waren, musste er sich zunächst mühsam einarbeiten. Es bedurfte zahlreicher Anläufe, um die App zum ersten Mal zu erstellen, da zum damaligen Zeitpunkt von Flutter noch keine vorgefertigten Build-Abläufe zur Verfügung standen.

Github Action Es war eine mühsame Aufgabe, Xcode zu konfigurieren, und nach mehreren Versuchen wurde schließlich die erste .ipa-Datei erstellt. Allerdings war der schwierigste Teil noch nicht überwunden, da das Ziel darin bestand, bei jedem GitHub-Push automatisch eine iOS-App zu erstellen.

Um über GitHub Actions eine iOS-App zu erstellen, sind mehrere Actions-Secrets erforderlich, die aus Xcode bezogen werden müssen. Folgende Secrets waren notwendig:

- **BUILD_CERTIFICATE_BASE64:** Dieses Secret enthält das Zertifikat, das zur Signierung der iOS-App verwendet wird. Es muss von einer vertrauenswürdigen Zertifizierungsstelle ausgestellt worden sein und wird normalerweise als Base64-codierte Zeichenfolge bereitgestellt.
- **BUILD_PROVISION_PROFILE_BASE64:** Dieses Secret enthält das Provisioning-Profil, das zur Installation der iOS-App auf einem Gerät oder für die Verwendung von Testflight verwendet wird. Das Provisioning-Profil enthält Informationen über die Geräte, auf denen die App installiert werden kann, sowie das verwendete Zertifikat. Auch dieses Secret wird normalerweise als Base64-codierte Zeichenfolge bereitgestellt.
- **KEYCHAIN_PASSWORD:** Dieses Secret ist das Passwort für den Zugriff auf den Schlüsselbund, in dem das Signaturzertifikat und der dazugehörige private Schlüssel gespeichert sind. Der private Schlüssel wird benötigt, um die App zu signieren, und das Passwort schützt den Zugriff auf den Schlüsselbund vor unbefugtem Zugriff.
- **P12_PASSWORD:** Dieses Secret ist das Passwort für das PKCS#12-Dateiformat, das das Signaturzertifikat und den privaten Schlüssel enthält. Diese Datei wird normalerweise für die Signierung von iOS-Apps verwendet. Das Passwort schützt die Datei vor unbefugtem Zugriff und stellt sicher, dass nur autorisierte Personen Zugriff auf den privaten Schlüssel haben.

Nach mehreren Versuchen, die Actions funktionsfähig zu machen, gelang es schließlich, eine .ipa-Datei zu erstellen.

Firebase App Distribution Für die Testphase und die Beta-Version der iOS-App nutzten wir, wie auch bei der Android-App, Firebase App Distribution. Beim Hochladen der .ipa-Dateien gab es allerdings Probleme mit den Zertifikaten.

Apple Developer Lizenz Um eine .ipa-Datei auf einem iOS-Gerät auszuführen, ist eine verifizierte .ipa-Datei erforderlich. Die aktuelle Action konnte die App jedoch nur mit der lokalen Entwicklerlizenz verifizieren, um sie im Emulator auf MacOS auszuführen. Wenn die App also auf einem anderen Gerät installiert werden sollte, war dies nicht möglich. Dies war auch der Grund, warum der Upload auf Firebase App Distribution nicht funktionierte.

Es gibt verschiedene Möglichkeiten, um eine .ipa-Datei zur Verfügung zu stellen, darunter:

- **App Store-Verteilung:** Die offizielle Methode zur Verteilung von iOS-Apps an eine globale Benutzerbasis. Apple prüft jede App, bevor sie im App Store veröffentlicht wird.
- **Ad-Hoc-Verteilung:** Die Ad-Hoc-Verteilung ermöglicht es Entwicklern, ihre Apps an eine begrenzte Anzahl von Personen außerhalb des App Stores zu verteilen. Entwickler müssen die UDIDs der Testgeräte manuell registrieren und eine Ad-Hoc-Provisioning-Datei erstellen, die zusammen mit der App an die Tester gesendet wird.
- **TestFlight-Verteilung:** TestFlight ist eine kostenlose Plattform von Apple, die es Entwicklern ermöglicht, ihre Apps an ausgewählte Tester zu verteilen. Jeder Tester lädt einfach TestFlight herunter, meldet sich mit seiner Apple-ID an und erhält Zugriff auf die App.
- **Enterprise-Verteilung:** Die Enterprise-Verteilung ermöglicht es Unternehmen, ihre Apps intern an Mitarbeiter oder Kunden zu verteilen, ohne den App Store zu nutzen. Sie benötigen jedoch eine spezielle Enterprise-Entwicklerlizenz von Apple und müssen die Apps auf einem eigenen Unternehmens-Server hosten.
- **B2B-Verteilung:** Die B2B-Verteilung ermöglicht es Entwicklern, ihre Apps speziell für Unternehmen und Organisationen zu entwickeln und diese direkt an diese Kunden zu verkaufen. Die Apps können dann über den App Store oder über ein spezielles B2B-Programm verkauft werden.
- **TestFlight für interne Tests:** TestFlight für interne Tests ist ähnlich wie die öffentliche TestFlight-Verteilung, jedoch speziell für interne Tests in Unternehmen. Diese Methode erfordert jedoch ein Apple Developer Enterprise-Programm.
- **Sideloadung:** Sideloadung ist eine Methode, bei der eine App von einem Drittanbieter auf einem iOS-Gerät installiert werden kann. Diese Methode erfordert jedoch, dass die Sicherheitseinstellungen des Geräts geändert werden und die App möglicherweise nicht von Apple genehmigt wurde.

Zunächst wurde die Methode des Sideloadings mittels des Tools Signulous ausprobiert, da sie als kostenfreie Option bekannt war. Dabei konnte die iOS-Anwendung erfolgreich installiert werden und manuelle Uploads auf Firebase App Distribution waren

ebenfalls möglich. Um jedoch eine Automatisierung zu ermöglichen, musste eine andere Lösung gefunden werden.

Da die HTL Leonding ihren Schülerinnen und Schülern Entwickler-Accounts zur Verfügung stellt, hat das Team um Zugriff auf einen solchen Account. Allerdings wurde dieser Account so eingeschränkt, dass es nicht möglich war, die Anwendung zu verifizieren. Nach weiteren Anfragen bezüglich einer Erweiterung der Berechtigungen für die Verifizierung wurde klar, dass dies nicht möglich war.

Somit blieb nur noch die Möglichkeit, einen kostenpflichtigen Entwickler-Account für \$99 pro Jahr zu erwerben. Das Team entschied sich jedoch dagegen, da der Aufwand bereits zu hoch war und weitere Zeit verschwendet würde.

Dennoch wurde versucht, auf TestFlight umzusteigen, was jedoch ebenfalls den Kauf eines Entwickler-Accounts erforderte. Sobald die Android-Version mit ihren Funktionen weiterentwickelt ist, plant das Team, die Arbeit an der iOS-Version wieder aufzunehmen, um die Anwendung im App Store zu veröffentlichen.

Build Android

4.1.2 Fastline

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.

Die 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[13][14], 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

Diagramm erklärung screenshots

Regestrierungs Flow

Diagramm erklärung screenshots

Firebase Authentifizierung

allg.

4.2.4 Feed

Die Feed-Page ist die erste und wichtigste Anzeige auf der App. Hier werden alle aktuellen Beiträge und Updates von der Nachbarschaft angezeigt.

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

Der Filter bietet die Option die Beiträge nach bestimmten Kriterien zu filtern und die Suche nach bestimmten Beiträge zu vereinfachen. Der Filter unterteilt sich in einen Menüfilter und Hauptfilter. Die Ansicht vom Menüfilter befindet sich direkt über den Beiträgen und ermöglicht eine schnelle Filterung nach einzig allein den Hauptkategorien. Die Ansicht vom Hauptfilter taucht erst nach dem Antippen vom Filtersymbol auf und beinhaltet eine größere Auswahl an Filteroptionen. Darunter zählt neben dem Filtern nach Hauptkategorien auch die zusätzliche Möglichkeit genauer nach Unterkategorien zu suchen. Außerdem besteht auch die Option die Beiträge nach Datum oder Likes zu sortieren oder die Beiträge aufsteigend oder absteigend zu ordnen. Die wichtigste Filterkomponente ist der Range-Slider, womit die Beiträge nach der Reichweite gefiltert werden können, da die Entfernung zum Nachbarn eine der wichtigsten Entscheidungsfaktoren zum Antworten auf einem Beitrag ist.

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**

Die Profilanzeige ist die öffentliche Informationsstelle über den User. In dieser Anzeige werden als erster Eindruck der Name und das Profilbild vom User angezeigt. Genauere Informationen über den User können im Bereich Nutzerinfo gefunden werden. Darunter zählen:

- Geburtstag
- Beruf
- Bio

Außerdem beinhaltet die Anzeige eine eigene Beitragssicht, wo alle Beiträge vom jeweiligen User eingesehen werden können.

Profil Melden

Falls das Profil vom User unangebrachten Inhalt aufweist, besteht die Möglichkeit das Profil zu melden.

4.2.12 Benachrichtigungen

Benachrichtigungen dienen dazu die Nachbarn über mögliche Hilfsbereitstellungen zu informieren, bevor der Kontakt überhaupt entsteht. Die Benachrichtigung zeigt den Nachbarn an, der in Kontakt treten möchte, und möglicherweise den Beitrag auf dem geantwortet wurde. Außerdem ist die Benachrichtigung mit einem „Annehmen“- und „Ablehnen“- Button ausgestattet, womit das Hilfsangebot angenommen oder abgelehnt werden kann.

4.2.13 Einstellungen

Die Einstellungssicht der Anwendung bietet eine Reihe an nützlichen Funktionen, die dem User mehr Kontrolle über sein Konto geben. Dazu gehört zu einem die Option, die Sprache der App umzustellen, um dem User zu ermöglichen, die Anwendung in der bevorzugten Sprache zu nutzen. Zum derzeitigen Stand kann die App sich in zwei Sprachen übersetzen lassen: Deutsch und Englisch. Darüber hinaus können Benutzer auch entscheiden, ob sie Benachrichtigungen erhalten möchten oder nicht, und diese Einstellungen jederzeit ein- oder ausschalten. Diese Einstellungen bieten den Benutzern eine höhere Privatsphäre und Personalisierungsmöglichkeiten, um die Anwendung besser an ihre individuellen Bedürfnisse anzupassen. Zu den weiteren Einstellungen gehört das Umändern der E-Mail oder des Passworts, um die Sicherheit des Kontos zu gewährleisten und unbefugten Zugriff zu verhindern. Die letzte Funktion in der Einstellungssicht ist das Löschen des eigenen Kontos, womit alle Daten des Users gelöscht werden. Vor dem endgültigen Löschen des Kontos wird der User allerdings aufgefordert, seine Entscheidung zu bestätigen.

4.2.14 Feedback

Im Zuge unserer Testphase suchen wir nach einer effizienten Methode, um Feedback zu sammeln. Um ein fundiertes Verständnis einer Fehlermeldung zu erlangen, erweist sich die Bereitstellung eines Screenshots als besonders nützlich. Daher haben wir uns nach Packages umgesehen, welche die Möglichkeit bieten, Screenshots zu erstellen und ergänzende Informationen als Kontext hinzuzufügen. Wir haben uns für das Package feedback entschieden. Dieses Package gestattet dem Nutzer, einen Screenshot anzufertigen und Annotationen hinzuzufügen, bevor ein entsprechender Kontext in Textform eingefügt werden kann. Eine weitere praktische Funktion besteht darin, dass der Nutzer bei Bedarf noch einmal navigieren und den Screenshot wiederholen kann. Zur Initiierung dieses Feedback-Views haben wir das Package shake genutzt, welches bei einer entsprechenden Bewegung des Mobilgeräts aktiviert wird. Wir haben uns bei dieser Funktion an anderen Anwendungen orientiert, die ähnliche Mechanismen für die Feedback-Erhebung nutzen.

Das von uns gewählte Vorgehen zur Analyse des Feedbacks erfolgt über die Trello-API, welche es uns ermöglicht, ein entsprechendes Dashboard zur Erfassung und Bearbeitung der Meldungen zu erstellen. Die erfassten Informationen, wie beispielsweise der Screenshot, der Text, das Datum und relevante Systeminformationen, dienen uns als Grundlage

für eine weitere Auswertung und Fehlerbehebung. Trello erleichtert uns die Verwaltung des Feedbacks auf kostenlose Weise und stellt somit ein adäquates Instrument zur Unterstützung des Entwicklungsprozesses dar.

4.3 UI/UX Design

Das Design der Nachbarschafts-App war für das Team von hoher Bedeutung, weshalb der Designer Martin Hausleitner erheblichen Zeit- und Arbeitsaufwand aufwendete, um ein anspruchsvolles Design zu gestalten. Angesichts der Tatsache, dass die App ein breites Publikum ansprechen soll, einschließlich verschiedener Altersgruppen, war es von entscheidender Bedeutung sicherzustellen, dass die Benutzer die App einfach und intuitiv bedienen können. Um dieses Ziel zu erreichen, wurde eine klare Struktur und Navigation in das Design integriert, um den Benutzern eine einfache und effektive Nutzung der gewünschten Funktionen zu ermöglichen. Es wurde darauf geachtet, dass die Buttons gut erkennbar und mit aussagekräftigen Icons und verständlichen Beschriftungen versehen sind, um Verwirrung zu vermeiden. Weiterhin wurde das Design der Karten auf eine organische Weise gestaltet, um ein harmonisches und ästhetisches Gesamtbild zu erzeugen.

4.3.1 Inspiration

Während des Designprozesses für die App wurden umfangreiche Recherchen im Bereich App-Design durchgeführt. Das Ziel war, die App so intuitiv wie möglich zu gestalten, um eine benutzerfreundliche Erfahrung sicherzustellen. Dabei wurden bekannte Social-Media-Apps wie Twitter, Instagram und TikTok als Orientierung genutzt, da diese bereits erfolgreich auf dem Markt etabliert sind und von vielen Menschen vertraut genutzt werden.

Zusätzlich diente die erfolgreichste Nachbarschafts-App in Deutschland, Nebenan, als Grundlage für die App-Entwicklung. Allerdings wurde festgestellt, dass ihre App sehr kompliziert aufgebaut und unübersichtlich ist. Daher wurde dies als Chance gesehen, um es besser zu machen.

Zur Entwicklung eines einfachen und schlichten Designs wurden Inspirationen von Websites wie Dribble und Mobbin genutzt. Insgesamt war die Recherche und Inspiration

für das Design der App ein wichtiger Schritt, um sicherzustellen, dass Benutzer eine ansprechende und intuitive Erfahrung haben.

4.3.2 Prototyping

Das Team legte von Anfang an großen Wert auf eine exzellente Benutzererfahrung, und daher war es ihnen klar, dass ein Prototyp gestaltet werden musste. Das Prototyping hatte für die Entwickler den großen Vorteil, dass sie beim Programmieren in Flutter nicht mehr lange darüber nachdenken mussten, wie Menüs oder Bildschirme gestaltet werden sollten. Somit war es einfach für Teammitglieder, die nicht mit Design vertraut waren, den Prototypen als Grundlage zu nutzen, um ihre Umsetzung in Flutter zu gestalten. Der Prototyp selbst war für den Designer Martin Hausleitner eine Reise durch verschiedene Designer-Tools, die im folgenden Absatz genauer behandelt werden.

Framer

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. Dies sind überzeugende Gründe für den Designer, den ersten Prototypen mit Framer zu gestalten.

Framer wurde während des Hackerthon Linz haCKt verwendet, um den ersten Prototypen zu erstellen wie bei der man bei der Abbildung 5 erkennen kann. 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

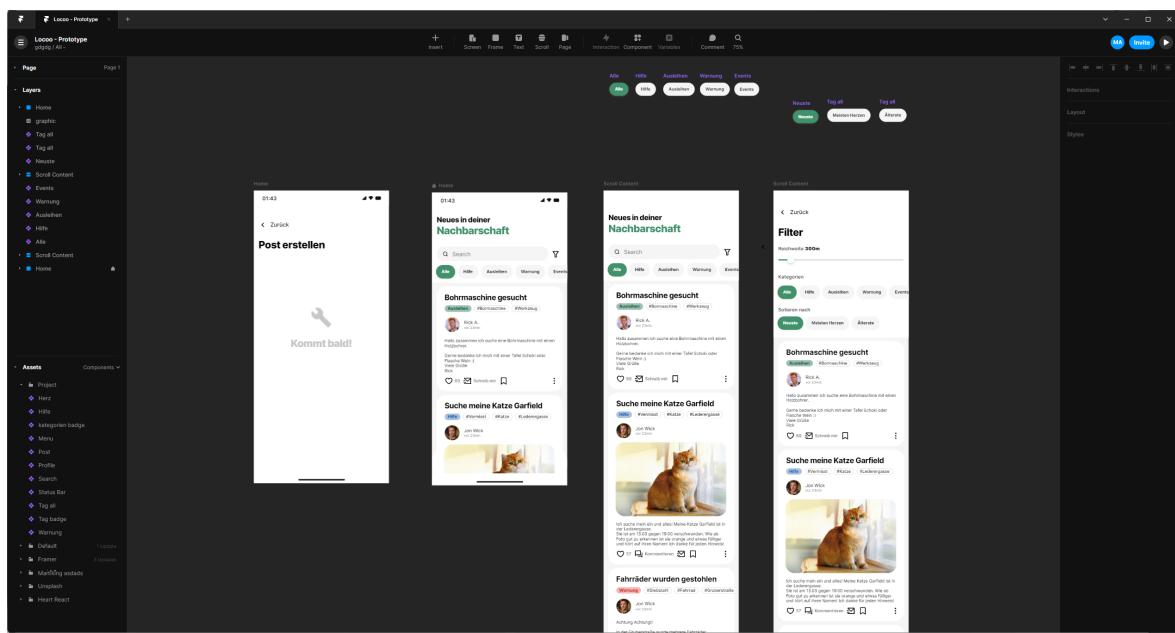


Abbildung 5: Nochba Prototyp in Framer

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

Bei der ersten Nutzung von Framer wurden zahlreiche Funktionen und Optionen entdeckt, was zunächst begeisterte. Doch mit der Zeit wurde erkannt, dass dies für das betreffende Projekt zu umfangreich war und somit nach einer einfacheren Lösung gesucht werden musste. Die Wahl fiel auf Adobe XD, da bereits jahrelange Erfahrung mit diesem Tool vorhanden war.

Trotz des geringeren Funktionsumfangs im Vergleich zu Framer ist Adobe XD aufgrund seiner Benutzerfreundlichkeit und Einfachheit ein ideales Werkzeug, um schnell Prototypen zu erstellen. Alle Hauptscreens wurden fertiggestellt wie man bei Abbildung 6 sehen kann, bevor die Entwicklung mit Flutter begann. Die anderen wichtigen Screens wurden später gestaltet, nachdem eine bessere Kenntnis von Flutter erlangt wurde und es möglich war, den Aufwand für die Umsetzung besser abzuschätzen.

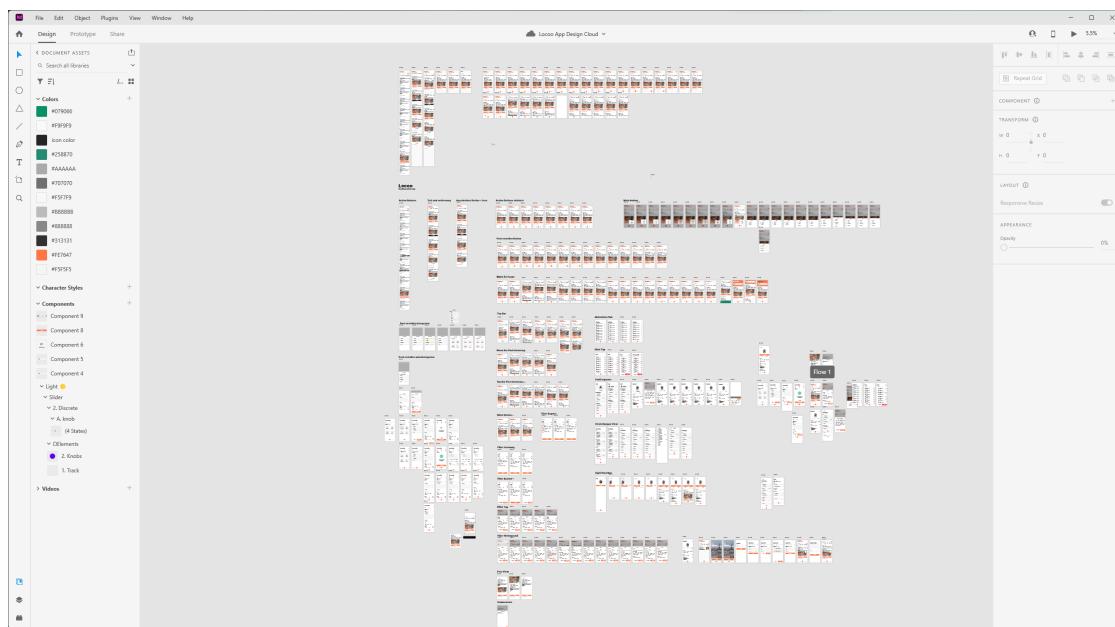


Abbildung 6: Nochba Prototyp in Adobe XD

Die Abbildung 6 zeigt eine Sammlung von Designs und Designentwicklungen in unserem Prototypen. Adobe XD war für unser Projekt geeignet, da Martin Hausleitner als alleiniger Designer keine Kollaborationsfunktion benötigte.

Jedoch ist der Designer mit dem bekannteren und kostengünstigeren Figma vertraut, das bei der Zusammenarbeit an einem Design und aufgrund der zusätzlichen Funktionen besser geeignet ist.

Zusammenfassend kann festgestellt werden, dass Adobe XD ein leistungsstarkes Tool für UI/UX-Designer ist, das einfach zu bedienen ist und für kleine bis mittelgroße Projekte geeignet ist. Obwohl es nicht so viele Funktionen wie andere Tools bietet, eignet es sich gut für die schnelle Erstellung von Prototypen und eine einfache Zusammenarbeit mit Entwicklern und Stakeholdern.

Design System

Der Designer Martin Hausleitner hat ein Design-System entwickelt, um bei der Gestaltung der App eine klare Struktur zu haben. Das System umfasst verschiedene UI-Elemente, von denen die wichtigsten in Abbildung 7 dargestellt sind.

Cards Ein häufig genutztes UI-Element in der App ist das Card-Design. Der Designer hat hierbei darauf geachtet, alle Elemente in einem abgerundeten, grauen Container mit hellem Grau zu gestalten, um einen guten Kontrast zum Hintergrund zu erzeugen.

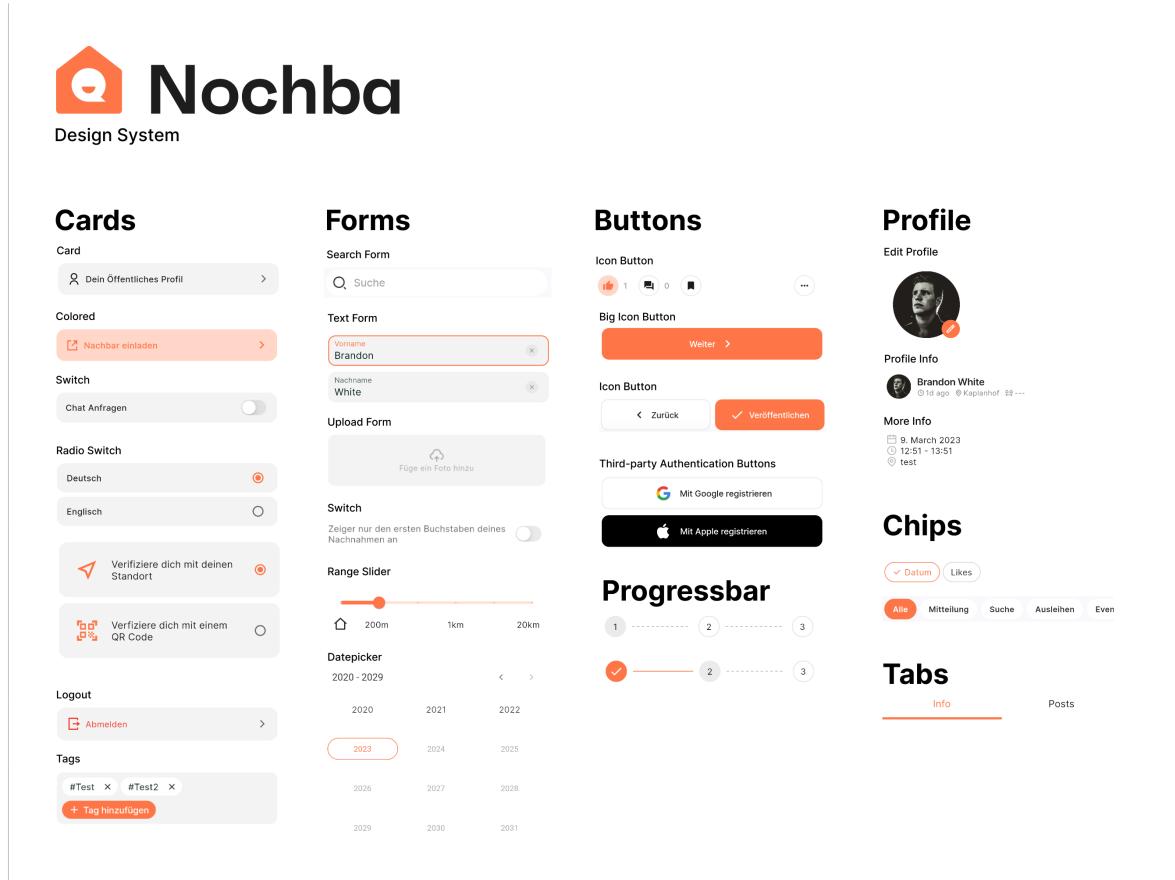


Abbildung 7: Nochba Design System

Um den Nutzer zur Interaktion aufzufordern, wurde beispielsweise eine Einladungskarte für Nachbarn gestaltet, die in den Primär- und Sekundärfarben gehalten ist.

Für die Switch Card wurde das Switch-Design des verfügbaren Cupertino UI-Widgets genutzt, da es dem Designer gut gefiel.

Für den Radio Switch gibt es eine größere und eine kleinere Variante, um verschiedenen Design-Anforderungen gerecht zu werden.

Der Logout-Button wurde bewusst mit einem roten Icon und roter Schrift gestaltet, um dem Nutzer deutlich zu machen, dass er sich ausloggt.

Besondere Aufmerksamkeit erforderte der Tag-Editor, der einer der aufwändigsten Widgets in Flutter war. Der Designer hat diesen komplett neu durchdacht, um das Erstellen und Abspeichern von Tags leicht verständlich zu gestalten, so dass jeder Nutzer damit umgehen kann.

Forms Das Suchformular wurde mit einem organisch runden Design gestaltet, das zum Feed-Screen der App passt. Die Formulare, die in der gesamten App verwendet

werden, wurden in Flutter sehr aufwändig gestaltet, um den Vorgaben des Designers zu entsprechen.

Um dem Nutzer eine klare Rückmeldung zu geben, wurde ein orangefarbener Ring um das angeklickte Feld platziert. Es wurde darauf geachtet, dass das Label des Textfeldes immer gut lesbar ist auch wenn Cursor innerhalb des Feldes ist. Zudem wurde ein X-Button hinzugefügt, der den gesamten Inhalt des Textfelds löscht.

Darunter befindet sich ein Upload-Formular, das im Design einer großen Card gestaltet ist, um dem Nutzer das Klicken zu erleichtern.

Der Switch, der in den Formularen verwendet wird, wurde aus dem iOS-Design übernommen.

Der Range-Slider wurde unter Verwendung des Material Designs gestaltet und mit den passenden Farben angepasst. Zusätzlich wurden Labels hinzugefügt, um dem Nutzer eine bessere Orientierung zu ermöglichen.

Für den Datepicker wurde die Bibliothek Syncfusion_flutter_datepicker verwendet. Das UI wurde entsprechend den Farben der App angepasst, um eine konsistente und ästhetische Benutzeroberfläche zu gewährleisten.

Buttons Die Icon-Buttons stellten das aufwändigste Design-Element dar, da es bei Flutter mehrere Möglichkeiten gibt, um sie zu implementieren. Dies führte dazu, dass sie häufig geändert werden mussten, um das richtige Design zu finden. Die runden Icon-Buttons ziehen sich durch die gesamte App.

Der Big Icon Button wurde häufig eingesetzt, da er die Primary Color aufweist und somit eine wichtige Funktion anzeigt. Dem Designer war es wichtig, dass jeder Button immer ein Icon hatte, um dem Nutzer die Bedienung zu erleichtern. Es wurde darauf geachtet, dass alle Buttons die gleiche Rundung aufweisen.

Für den normalen Icon-Button wurde ein dezenter grauer Hintergrund verwendet, um ihn als Secondary Button zu kennzeichnen. Auch hier wurden immer Icons verwendet.

Da wir in unsere App auch Third-Party-Anmeldemöglichkeiten hinzufügen wollten, haben wir auch das Login mit Apple und das Login mit Google an das Design angepasst, um eine einheitliche Benutzeroberfläche zu gewährleisten.

Die Progressbar wurde beim Registrierungsprozess und beim Erstellen eines Posts eingesetzt, um dem Nutzer eine bessere Orientierung zu geben, wie viele Schritte

noch zu erledigen sind. Ursprünglich war geplant, das Design von Material zu nutzen. Aufgrund der Einschränkungen bei der Umsetzung des Designs mit dem Material-Design in Flutter und den spezifischen Anforderungen des Designers, war es erforderlich, die Progressbar komplett in Flutter zu gestalten.

Profile Der Edit Profile-Button wurde beim Registrieren und Bearbeiten eines Profils verwendet, um das Profilbild zu ändern. Es wurde darauf geachtet, dass der Button auffällig ist, damit der Nutzer ihn schnell erkennen kann.

Die Profil-Informationen werden bei jedem Beitrag angezeigt, darunter die Zeit, in welchem Stadtteil der Nachbar wohnt und die ungefähre Entfernung zu ihm. Es wurde darauf geachtet, dass die Informationen übersichtlich mit Icons dargestellt werden, um dem Nutzer eine schnelle Orientierung zu ermöglichen.

More Info wird beispielsweise bei einem Event-Eintrag oder den persönlichen Daten eines Profils verwendet. Hier wurden Icons verwendet, um dem Nutzer eine einfache und verständliche Darstellung zu ermöglichen.

Chips Chips wurden in unserer App oft verwendet, um Filter auszuwählen. Der Designer war mit dem finalen Design jedoch noch nicht vollständig zufrieden, da es viele unterschiedliche Varianten gibt. Es wird weiterhin an der Gestaltung der Chips gearbeitet, um eine einheitliche und ästhetisch ansprechende Darstellung zu erreichen.

Tabs Die Implementierung von Tabs war auch in Flutter eine Herausforderung, da die Positionierung nicht der Standard-Tabs-Position entsprach. Aus diesem Grund wurde die Gestaltung aufwändig gestaltet, um ein optimales Ergebnis zu erzielen. Die Mühe hat sich jedoch gelohnt, da die Tabs nun eine gute funktionale Benutzeroberfläche bieten.

Farben

Die Farbpalette einer Marke spielt eine wichtige Rolle im UI-Design, da sie dazu beiträgt, ein konsistentes Erscheinungsbild zu schaffen. Bei der Gestaltung einer Nachbarschafts-App wurde eine Vielzahl von Farbpaletten von anderen Apps untersucht. Es wurde festgestellt, dass die meisten großen Apps die Farbe Grün verwenden.

Obwohl Grün oft mit Natur und Gemeinschaft assoziiert wird, wurde sich bewusst dazu entschieden, sich von diesen etablierten Konventionen abzuheben. Stattdessen wurde die Farbe Orange ausgewählt, da sie auffällig und ungewöhnlich ist und somit das Potenzial hat, die App von anderen Nachbarschafts-Apps zu unterscheiden.

Zusätzlich passt die Farbe Orange gut zu den Werten der App, da sie für Wärme, Freundschaft und Optimismus steht - Eigenschaften, die in der App gefördert werden sollen. Die Hoffnung besteht darin, dass die Farbauswahl dazu beitragen wird, dass die Nutzer sich in der App wohl und willkommen fühlen und die Farbe sich als wiedererkennbares Markenzeichen etabliert.

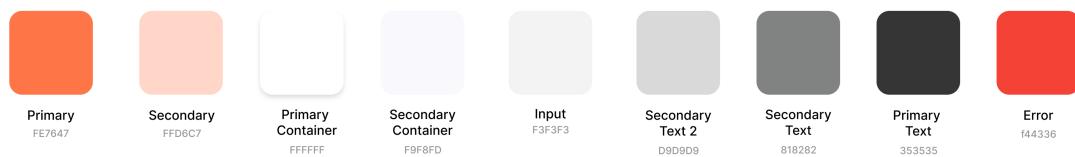


Abbildung 8: Nochba Farbpalette

Bevor der endgültige Farbton ausgewählt wurde, wurden verschiedene orangefarbene Töne für den Prototypen ausprobiert und für einige Tage beobachtet, um ein Gefühl für die Farbe zu bekommen. Schließlich wurde sich für die endgültigen Farben entschieden, wie in der Abbildung 8 dargestellt.

Die Hauptcontainerfarbe für die meisten Ansichten ist Weiß, ebenso wie die Hintergrundfarbe der Posts. Allerdings wurde eine Farbe gesucht, die gut als Hintergrund passt, weshalb ein helles Grau verwendet wurde.

Für die Textfarbpalette wurde fast Schwarz gewählt, da dies zu einem weicheren Eindruck führt. Um Untertitel weniger präsent zu gestalten, wurde ein etwas hellerer Grauton gewählt.

Icons

Remixicon



Material Icons



Eigen designtes Icon



Abbildung 9: App Icons

Die obige Abbildung 9 zeigt alle Icons, die in einer App verwendet werden. Das Ziel war es, eine möglichst große Anzahl von Icons zu integrieren, um die Benutzerfreundlichkeit und Bedienbarkeit der App zu verbessern. Nachdem verschiedene Icon-Packs betrachtet wurden, wurde hauptsächlich das frei verfügbare Remixicon-Pack verwendet, um der App ein einzigartiges Aussehen zu verleihen und sie von anderen Apps abzuheben. Das runde Design der Icons wurde besonders geschätzt. Ein weiterer Vorteil von Remixicons ist, dass es ein Flutter-Package gibt, was die Implementierung erleichtert hat.

Obwohl Remixicons viele Icons zur Verfügung stellt, wurden bei einigen Icons die Material Icons von Google bevorzugt. Insbesondere die abgerundeten Icons wurden verwendet, um das Design insgesamt weicher zu gestalten. Da kein passendes Icon vorhanden war, um den Abstand zwischen zwei Nachbarn zu symbolisieren, wurde ein neues Icon im Stil der anderen entworfen.

Typografie

Die Schriftfamilie Inter

Inter ist eine Schrift, die sorgfältig für Computerbildschirme entworfen und hergestellt wurde.

ABCDEF^GH^IJ^KL^MN
OPQRSTU^VW^XYZ
abcdefghijklmnopq
rstuvwxyz () & ?! @
1234567890 .,: /→

Size **16** dp = spacing **-0.011 em**

Es gibt natürlich kein absolutes Richtig oder Falsch, wenn es darum geht, sich mit Typografie auszudrücken, aber Inter Dynamic Metrics bietet Richtlinien für gute Typografie. Sie geben einfach die optimale Schriftgröße an, und die Laufweite und der Zeilenabstand werden für Sie berechnet, um die besten Ergebnisse zu erzielen.

Inter Bold Sample

**A B C D E F G G
H I J K L M N O P
Q R S T U V W X
Y Z 1 2 3 4 5 6 7
8 9 0 a å b c d e
f g h i j k l m n o p
q r s t u v w x y z.**

Inter Regular Sample

Dynamic Metrics

The method by means of which the musical signals may be sent simultaneously in both directions along the same circuit is shown in our next illustration, figures 8, 9, and 10. The arrangement is similar to that shown in figures 3, 4, and 5, excepting that the primary wire of the transmitting instruments is passed through the primary wires of an induction coil, and the receiving instruments are placed in circuit with the secondary wire. In this way free earth communication is obtained between the two stations. It was also found by practical experiment that a key could be manipulated at either end of the circuit, and the musical signals produced by the manipulation of any key are received at all the stations upon the line.

The method by means of which the musical signals may be sent simultaneously in both directions along the same circuit is shown in our next illustration, figures 8, 9, and 10. The arrangement is similar to that shown in figures 3, 4, and 5, excepting that the primary wire of the transmitting instruments is passed through the primary wires of an induction coil, and the receiving instruments are placed in circuit with the secondary wire. In this way free earth communication is obtained between the two stations. It was also found by practical experiment that a key could be manipulated at either end of the circuit, and the musical signals produced by the manipulation of any key are received at all the stations upon the line.

Abbildung 10: Die Schriftfamilie Inter

Ein wichtiger Teil jedes Designs ist die Typografie, so auch bei der Nnochba-App.

Der Designer der App war auf der Suche nach einer Schriftart, die modern, ansprechend und gut lesbar ist. Er suchte nach einer Schriftart, die Lesbarkeit und das Verständnis des Textes verbessert. Schließlich stieß er auf die Schriftart "Inter" von Rasmus Andersson [15], die man auf der Abbildung 10 betrachten kann.

Inter ist eine moderne und ansprechende Schriftart, die speziell für die Verwendung auf Bildschirmen entwickelt wurde. Die Schriftart ist in vielen verschiedenen Stilen verfügbar und bietet eine breite Unterstützung für verschiedene Sprachen und Schriftsysteme. Dies war besonders wichtig für den Designer der Nachbarschafts-App, da die App von Menschen aus verschiedenen Ländern genutzt wird, die möglicherweise unterschiedliche Sprachen und Schriftsysteme verwenden.

Ein weiterer Vorteil von Inter ist seine Lesbarkeit. Die Schriftart ist gut lesbar, auch in kleineren Größen, was besonders wichtig ist, da die App oft auf mobilen Geräten verwendet wird. Die klare und prägnante Schriftart von Inter verbessert auch die Lesbarkeit und das Verständnis des Textes, was wiederum zu einer besseren Benutzererfahrung führt.

Schließlich ist Inter auch eine Open-Source-Schriftart, die kostenlos verwendet werden kann [15]. Dies ist besonders wichtig für das Projekt Nnochba, da Nnochba für Open Source stehen will.

Zusammenfassend lässt sich sagen, dass Inter eine ausgezeichnete Wahl für die Nachbarschafts-App ist. Die Schriftart ist modern, ansprechend und gut lesbar, und bietet eine breite Unterstützung für verschiedene Sprachen und Schriftsysteme. Die Verwendung von Inter hat dem Designer der App geholfen, eine professionelle und kosteneffektive Lösung für die Entwicklung der App zu finden.

Logo

Logo Historie Im Zuge des Designprozesses wurden hohe Ansprüche an das Logo der App gestellt, da es das Projekt repräsentiert und es dem Team wichtig ist. Es war wichtig, dass das Logo einfach gehalten und leicht erkennbar ist. Die endgültige Gestaltung des Logos hat viel Zeit in Anspruch genommen, da die vorherigen Entwürfe nicht vollständig zufriedenstellten die man bei Abbildung 12 sieht. Nach anderen Logos, die eine Verbindung zu Nachbarschaften oder Häusern herstellen, wurde gezielt gesucht

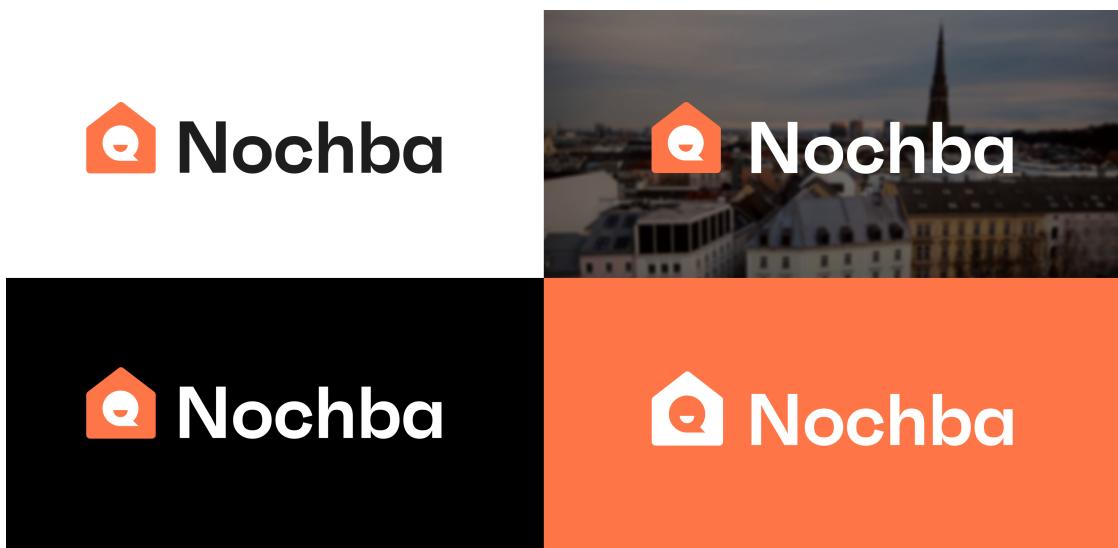


Abbildung 11: Finales Logo



Abbildung 12: Logo Design Ideen

und Ideen auf dem Discord-Server gespeichert. Es sollte immer ein Haus erkennbar sein, da Häuser schnell mit Nachbarschaften in Verbindung gebracht werden.

In den früheren Versionen der App gab es Schwierigkeiten, ein ansprechendes App-Icon zu gestalten, da kein eigenständiges Icon vorhanden war. Aus diesem Grund wurde entschieden, den Schriftzug und das App-Icon (Logo) separat zu gestalten, wie es auch bei anderen Apps üblich ist. Das endgültige Logo wurde erst Ende 2022 entworfen, da alle anderen Designs nicht gefielen. Es wurde ein Haus mit einer sprechenden Sprechblase, die lächelt, als Logo wie bei Abbildung 11 zu sehen ist gewählt. Dies soll symbolisieren, dass man innerhalb des Hauses sprechen kann und das Lächeln soll verdeutlichen, dass man Freude mit seinen Nachbarn teilen kann. Eine runde, verspieltere Schriftart wurde bewusst gewählt, da sie besser mit der Farbmischung harmoniert als die vorherige Schriftart.

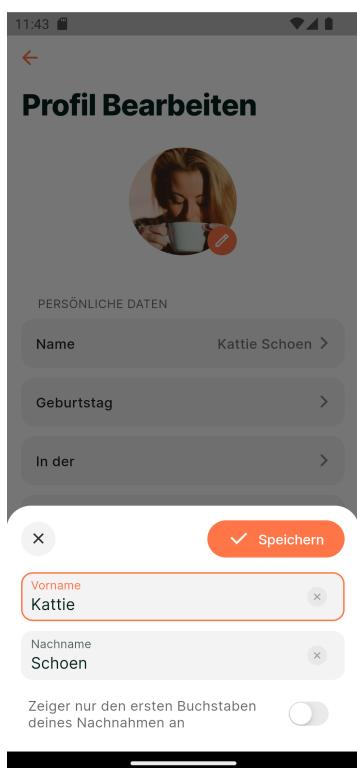


Abbildung 13: Screenshot des "Name bearbeiten"-Screens

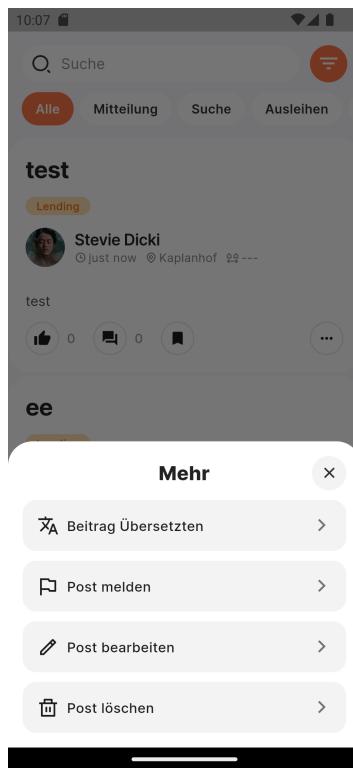


Abbildung 14: Screenshot des "Mehr"-Menüs für Beiträge

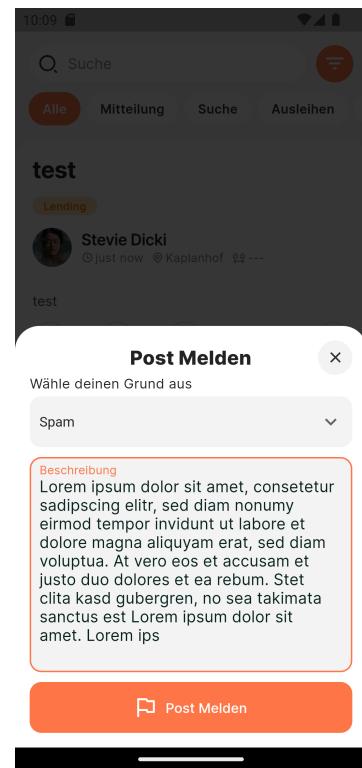


Abbildung 15: Screenshot des "Beitrag melden"-Menüs

4.3.3 App Design

Thumb Zone Prinzip

Im Design-Muster wurde versucht, das Thumb Zone Prinzip zu implementieren, indem wichtige UI-Elemente immer im unteren Bereich der App platziert wurden. Das Layout für die Namensbearbeitung 13 wurde extra unten platziert, um alle Klick-Elemente bequem mit dem Daumen bedienen zu können. Wichtige wie "Speichern" wurden in der primären Farbe gestaltet und rechts ausgerichtet, um einen leichteren Zugriff zu gewährleisten. Das Textfeld kann einfach durch Drücken der Enter-Taste auf der Tastatur geändert werden, um das Eintippen von Daten zu erleichtern.

Wie bei Abbildung 14 15 zu sehen ist, erfolgt eine weitere Implementierung einer Bottom View, um dem Thumb Zone Prinzip gerecht zu werden.

Unwichtige Buttons wurden in grau gestaltet, wie z.B. der X-Button in der Abbildung. Ein Löschen-Button neben dem Textfeld wurde eingebaut, der es dem Benutzer ermöglicht, den gesamten Text bequem zu löschen. Jedoch war es nicht möglich, diese Prinzipien vollständig umzusetzen, da die Zeit gegen Ende knapp wurde. In zukünf-

tigen Versionen der App soll sichergestellt werden, dass diese Prinzipien einheitlich angewendet werden.

Design Historie

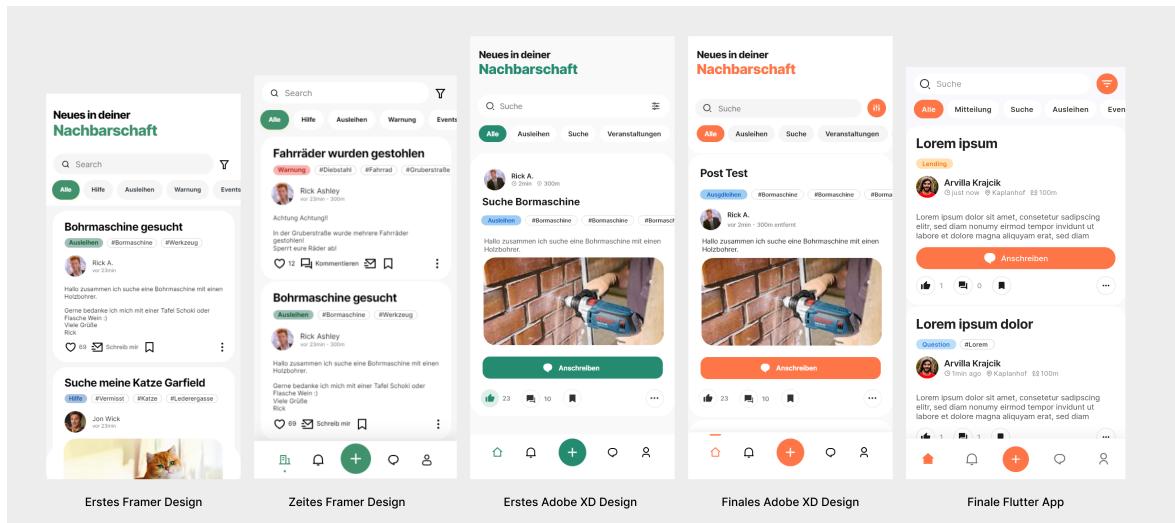


Abbildung 16: Screenshots verschiedener Versionen des App-Home-Screen

Im Rahmen des Hackathons Linz haCKt wurde das erste Design in Abbildung 16 ganz links erstellt. Hierbei wurden mehrere Brainstorming-Meetings mit Mentoren abgehalten, um ein Grundkonzept in Adobe XD zu gestalten. Anschließend wurden die groben Züge des Designs in Absprache mit dem Team festgelegt. Ein funktionsfähiger Prototyp wurde in Framer erstellt und am Tag der Endpräsentationen vorgestellt und verlinkt. Nach Linz haCKt wurde das zweite Design in Framer entwickelt.

Zwischen dem zweiten Framer-Design 16 und dem ersten Adobe XD-Design 16 wurde bewusst kein Abstand am Rand eines Beitrags gehalten, um den vollen Handy-Display auszunutzen. Der Filter-Button wurde mit der Suchleiste verbunden, um das Design stimmiger zu gestalten. Bei den Profilinformationen wurde auf Icons umgestellt, um es für neue Nutzer verständlicher zu machen. Diese Entscheidung wurde kurzzeitig verworfen, jedoch später in der Flutter-App wieder eingebaut.

Wie man erkennen kann, wurde für das Finale Adobe XD-Design 16 eine orangene Farbe für die Kapitelfarben gewählt. Der letzte Screenshot zeigt die Flutter-App, wie sie im Play Store erhältlich ist. Die größte Änderung gegenüber dem vorherigen Design betrifft den Hintergrund bei der Suche und der Kategorieauswahl. Der Hintergrund wurde hellgrau gemacht, um die Beiträge besser hervorzuheben. Bei der Entwicklung mit Flutter stellte sich heraus, dass ein Strich über dem aktiven Icon sehr aufwändig zu implementieren ist, weshalb darauf verzichtet wurde.

4.4 Website

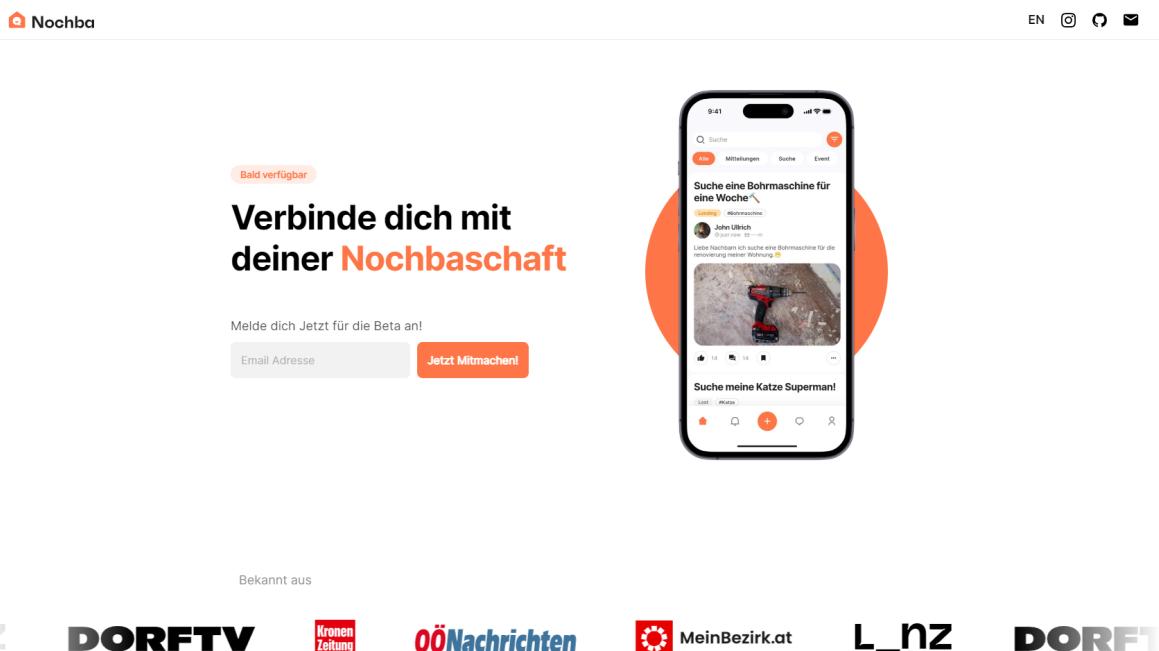


Abbildung 17: Screenshot der Nochba Website: nochba.at

Aufgrund der Medienpräsenz, die unser Projekt durch unsere Teilnahme an Linz hACkT erfahren hat und der Tatsache, dass wir keinen zentralen Anlaufpunkt für Informationen hatten, haben wir uns dazu entschlossen, eine schlichte Landing Page mit grundlegenden Informationen unter nochba.at zu gestalten.

Unsere Teilnahme an dem mPreneur Social Mobile Entrepreneurship von Arsham Edalatkhah hat uns zu einer internationalen Aufmerksamkeit verholfen. Dies hat uns dazu bewogen, eine englischsprachige Website unter nochba.com zu erstellen, um unsere Arbeit einem breiteren Publikum zugänglich zu machen und unsere Reichweite zu erhöhen.

4.4.1 Beta-tester anmeldung

Das wichtigste Merkmal der Website ist die Option für Nutzer, sich für die Testphase zu registrieren. Ursprünglich wurde versucht, die E-Mail-Adressen der Nutzer über die Google Sheets API zu speichern. Allerdings erwies sich dieser Ansatz als ineffektiv und dauerte mehrere Stunden. Als Alternative wurde das Framer-Add-On von Mailchimp.com genutzt, um Zeit zu sparen. Der kostenlose Plan von Mailchimp war für die Bedürfnisse ausreichend. Stand 7.3.2023 wurden 34 Beta-Anmeldungen über das Formularfeld gesammelt.

4.4.2 Design

Für die Gestaltung der Landing Page ließ sich der Designer von den Vorlagen von Framer inspirieren und nutzte vorgefertigte Abschnitte. Diese wurden jedoch so modifiziert, dass sie dem Designsysten gerecht werden. Die Farbpalette wurde beibehalten und es wurde darauf geachtet, dass die Website möglichst einfach gestaltet ist.

4.4.3 Kontent

Auf der Website sind folgende Informationen abgebildet:

- Zeitungsartikel
 - Mein Bezirk
 - Linz
 - OÖNachrichten
 - DorfTV
 - Kronen Zeitung (nur auf Papier)
- Unsere Mission
- Beitrag Kategorien
- Auszeichnungen
 - Immotopia Innovation Award
 - mPreneur Austria
 - Linz hACkt
- Features der App
- Partner der Diplomarbeit
- Über uns
- Links
 - Github Repo
 - Instagram
 - Email project@nochba.com

4.4.4 Hosting

Die Website wurde mithilfe von Framer erstellt und auf deren Hosting-Plattform gehostet. Um die Domain-Namen zu nutzen, die über GoDaddy (nochba.at und nochba.com) gekauft wurden, wurde eine Weiterleitung mit Maskierung auf die entsprechende Framer-URL eingerichtet. Durch diese Maßnahme wurde im Browser bei der URL die gekaufte Domain angezeigt.

SSL-Zertifikat

Leider hat die Website derzeit kein SSL-Zertifikat, da durch die Maskierung das Framer SSL-Zertifikat verloren geht. Allerdings wird die Website im Browser nicht als bedrohlich angezeigt, weshalb wir uns entschieden haben, kein SSL-Zertifikat zu erstellen, um den Aufwand zu minimieren.

4.5 Backend

4.5.1 Firebase

Firebase ist ein Backend-as-a-Service (BaaS), das Entwicklern eine enorme Erleichterung bei der Arbeit bietet. Das Hosting von Datenbanken und Cloud-Funktionen kann mit nur wenigen Klicks erfolgen, was die Notwendigkeit von Skalierung oder Ausfallvermeidung eliminiert. Das Firebase-Backend wird auf der Google Cloud in Frankfurt (EUR3 Europe-West) gehostet, um eine niedrige Latenz zu gewährleisten. Der kostenlose Spark-Plan war für den Gebrauch ausreichend, als der Dienst noch nicht die Firebase-Cloud-Funktionen genutzt hat. Um Kosten zu sparen, wurde lange Zeit mit dem Firebase-Emulator an den Cloud-Funktionen gearbeitet. Im Januar 2023 wurde auf den Blaze-Plan umgestiegen, um die Firebase-Cloud-Funktionen nutzen zu können. Bis März 2022 wurden nur wenige Euro für den Verbrauch gezahlt. Die Dokumentation und Tutorials von Firebase sind exzellent und es gibt viele Ressourcen auf YouTube.

4.5.2 Firebase SDKs

Firebase SDK ist eine Sammlung von Software Development Kits, die von Firebase bereitgestellt werden, um Cloud-basierte Anwendungen zu erstellen. Firebase SDK ist für eine Vielzahl von Programmiersprachen verfügbar, darunter JavaScript, Swift, Kotlin, Java und unter anderem Dart von Flutter. Die Firebase SDK bietet eine breite Palette

von Funktionen, mit denen leistungsstarke Anwendungen erstellt werden können, darunter Authentifizierung, Cloud Messaging, Cloud Firestore, Realtime Database, Cloud Functions. Mit diesen Funktionen können schnell und einfach Funktionen wie Benutzerverwaltung, Datenverwaltung, Messaging und Benachrichtigungen in Anwendungen integriert werden. Außerdem gibt es zu Firebase SDK eine umfangreiche Dokumentation und Support-Tools.

4.5.3 Firebase Authentication

Email und Passwort authentication

4.5.4 Cloud Firestore

NoSQL-Database, Dokument-basierte Speicherung, Subcollections

4.5.5 Cloud Storage

Speicherung von Profilbilder und Post-Bilder

4.5.6 Firebase Cloud Functions

Firebase Cloud Functions sind eine großartige Möglichkeit, um Business-Logik abzubilden. Es handelt sich um serverlose Funktionen, die in JavaScript oder TypeScript geschrieben werden und einzeln auf der Google Cloud gehostet werden. Je nach Nachfrage werden sie automatisch skaliert oder komplett abgeschaltet. Ein Nachteil ist, dass die Startup-Zeit länger sein kann als bei einem herkömmlichen Backend, das immer läuft. Allerdings kann man durch Cloud Functions viel Geld sparen, da nur die Prozessorlaufzeit bezahlt werden muss.

Firebase Cloud Functions ermöglichen es Entwicklern, auf verschiedene Ereignisse in Firebase-Produkten zu reagieren. Zum Beispiel, wenn sich Daten in der Firestore-Datenbank ändern oder ein neuer Nutzer in Firebase Authentication registriert wird. Wenn ein solches Ereignis eintritt, wird die entsprechende Cloud Function automatisch ausgeführt.

Wir haben alle Cloud Functions in TypeScript programmiert, da es Typsicherheit und erweiterte Fehlererkennung bietet.

Registrierung mit einem Verifizierungscode

Die Cloud-Funktion `checkVerificationCode` wird ausgeführt, wenn ein Nutzer sich mit einem Verifizierungscode registrieren möchte. Wenn der Benutzer nicht authentifiziert ist, wird eine `HttpsError` ausgelöst. Dann wird der Verifizierungscode überprüft, um sicherzustellen, dass er die korrekte Formatierung hat. Wenn der Code ungültig ist, wird eine weitere `HttpsError` ausgelöst.

Als Nächstes wird der Verifizierungscode mit der Datenbank abgeglichen, um sicherzustellen, dass er aktiv und noch nicht zu oft verwendet wurde. Wenn der Code erfolgreich validiert wird, wird die Adresse des Benutzers abgerufen und deren Koordinaten mithilfe einer API call an OpenStreetMap mit der Funktion `getOSMCoordinatesFromAddress` ermittelt. Dann wird die Entfernung zwischen der Adresse des Benutzers und der Adresse, die dem Verifizierungscode zugeordnet ist, berechnet mit der Funktion `getDistanceFromLatLonInMeters`. Wenn die Entfernung nicht innerhalb des zulässigen Bereichs liegt, wird eine weitere `HttpsError` ausgelöst.

Schließlich werden die Informationen des Benutzers und des Verifizierungscodes in der Datenbank aktualisiert, um anzusehen, dass der Benutzer erfolgreich verifiziert wurde. Die Cloud-Funktion gibt `true` zurück, um anzusehen, dass die Verifizierung erfolgreich war.

In jeder Phase der Funktion wird ein Logger verwendet, um Informationen über den Status der Funktion zu protokollieren.

Regestrierung mit Gerät Koordinaten

Die Cloud-Funktion `checkAddressWithDeviceLocation` erfordert eine authentifizierte Anfrage und erhält eine Adresse sowie Längen- und Breitengradkoordinaten vom Gerät des Benutzers. Die Funktion prüft, ob alle erforderlichen Daten vorhanden sind und ruft dann die Funktion `getOSMCoordinatesFromAddress` auf, um die Koordinaten der angegebenen Adresse zu erhalten. Es wird auch die Funktion `getDistanceFromLatLonInMeters` aufgerufen, um die Entfernung zwischen der Adresse und den Koordinaten des Geräts des Benutzers zu berechnen.

Wenn die Entfernung größer ist als ein vordefinierter maximaler Abstand, wird eine Fehlermeldung ausgegeben und die Funktion gibt `false` zurück. Andernfalls speichert die Funktion die Koordinaten der Adresse und die Entfernung zwischen den Koordinaten des Geräts des Benutzers und der Adresse in der Firestore-Datenbank. Die Funktion

ruft auch die Funktion `getOSMSuburbFromCoords` auf, um den Vorort der Adresse zu erhalten, und speichert diesen ebenfalls in der Firestore-Datenbank.

Die Funktion gibt `true` zurück, wenn die Verifizierung erfolgreich abgeschlossen ist.

In jeder Phase der Funktion wird ein Logger verwendet, um Informationen über den Status der Funktion zu protokollieren.

Verifizierungscode generieren

Die Cloud-Funktion `generateVerificationCode` definiert Konstanten für das Intervall zwischen der Generierung von Codes, die maximale Anzahl von Codes und die Reichweite in Metern.

Anschließend wird der letzte Code des Nutzers aus der Datenbank geholt, um zu überprüfen, ob seit dem letzten generierten Code ausreichend Zeit vergangen ist. Wenn ja, wird der zuletzt generierte Code zurückgegeben.

Wenn nicht, wird eine Schleife gestartet, um einen neuen Code zu generieren mit der Funktion `generateRandomVerificationCode`. Der generierte Code wird dann mit Firestore abgeglichen, um sicherzustellen, dass er nicht bereits verwendet wurde.

Wenn der generierte Code eindeutig ist, wird überprüft, ob der Benutzer verifiziert ist. Wenn dies nicht der Fall ist, wird ein Fehler zurückgegeben. Andernfalls wird der generierte Code in die Firestore-Datenbank eingefügt, um den letzten generierten Code und das Datum der Generierung zu speichern.

Das Skript gibt dann den generierten Code zurück. In jeder Phase der Funktion wird ein Logger verwendet, um Informationen über den Status der Funktion zu protokollieren.

Koordinaten von einer Adresse bestimmen

Ein wichtiger Teil für die Verifizierung ist, dass wir die Adresse des Nutzers in Koordinaten umwandeln, damit wir im Backend damit arbeiten können. Zunächst haben wir die Google Geocode API genutzt, allerdings kostet diese Geld. Wir haben eine bessere Option gefunden, die besser zu unserer Projekt-Philosophie passt: nämlich Nominatim, eine Open-Source-Geocoding-API für OpenStreetMap-Daten. Nominatim bietet eine kostenlose API für genau unser Problem an.

Die Funktion `getOSMCoordinatesFromAddress` nutzt die Bibliothek `axios`, um eine Anfrage an die Nominatim-API zu senden. Die API wird genutzt, um Koordinaten für eine Adresse zu erhalten.

Die Funktion nimmt einen Parameter `address` vom Typ `string` entgegen, welcher die Adresse enthält, für die Koordinaten abgerufen werden sollen.

Mithilfe von `axios.get` wird eine HTTP GET-Anfrage an die Nominatim-API gesendet. Die Adresse wird als URL-Parameter im Format `q=Adresse` übergeben. Die API liefert die Koordinaten als JSON-Objekt zurück, welches in der Variable `data` gespeichert wird.

Die Funktion überprüft dann, ob die Anfrage ein Ergebnis zurückgeliefert hat. Falls nicht, wird eine Fehlermeldung mit der Adresse ausgegeben.

Wenn ein Ergebnis vorhanden ist, wird das erste Ergebnis (in der Variable `firstResult`) verwendet, um eine neue Instanz der `GeoPoint`-Klasse zu erstellen. Diese Klasse ist Teil der Firebase-Admin-Bibliothek und ermöglicht die Speicherung von geografischen Koordinaten in einer Firestore-Datenbank.

Entfernung von zwei Nutzern berechnen

Die Cloud-Funktion `getDistanceFromTwoUsers` berechnet die Entfernung zwischen zwei Benutzern. Die Funktion erwartet eine `PostId` und einen authentifizierten Benutzer. Dann wird eine Überprüfung durchgeführt, ob der Post mit der angegebenen ID existiert und ob der Post eine gültige Reichweite hat. Es werden auch Überprüfungen durchgeführt, ob die Benutzerkoordinaten vorhanden sind, und ob die Entfernung zwischen beiden Benutzern innerhalb des Postbereichs liegt.

Die Funktion nutzt die importierten Funktionen `getDistanceFromLatLonInMeters` und `getNearestDistance`, um die Entfernung in Metern zu berechnen. Allerdings wird die berechnete Distanz grob gerundet, um die Privatsphäre der Nutzer zu wahren. Dabei werden die Längen- und Breitengradkoordinaten von zwei Benutzern miteinander verglichen, die aus der Firestore-Datenbank abgerufen werden. Wenn die Entfernung größer als die Reichweite des Posts ist, wird ein Fehler ausgegeben.

Schließlich gibt die Funktion die Entfernung zurück.

Entfernung von zwei geographischen Koordinaten berechnen

Listing 2: getDistanceFromLatLonInMeters Funktion

```

1     export function getDistanceFromLatLonInMeters(
2         lat1: number,
3         lon1: number,
4         lat2: number,
5         lon2: number
6     ) {
7         if (lat1 < -90 || lat1 > 90 || lon1 < -180 || lon1 > 180) {
8             throw new Error(
9                 "Invalid coordinate: lat1 must be between -90 and 90, lon1 must be
10                between -180 and 180"
11             );
12         if (lat2 < -90 || lat2 > 90 || lon2 < -180 || lon2 > 180) {
13             throw new Error(
14                 "Invalid coordinate: lat2 must be between -90 and 90, lon2 must be
15                between -180 and 180"
16             );
17         const R = 6371; // Radius der Erde in km
18         const dLat = deg2rad(lat2 - lat1); // deg2rad unten
19         const dLon = deg2rad(lon2 - lon1);
20         const a =
21             Math.sin(dLat / 2) * Math.sin(dLat / 2) +
22             Math.cos(deg2rad(lat1)) *
23                 Math.cos(deg2rad(lat2)) *
24                 Math.sin(dLon / 2) *
25                 Math.sin(dLon / 2);
26         const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
27         const d = R * c * 1000; // Distanz in Metern
28
29         if (lat1 === lat2 && lon1 === lon2) return 0;
30     }
31
32     function deg2rad(deg: number) {
33         return deg * (Math.PI / 180);
34     }
35 }
```

Der vorliegende Code implementiert eine Funktion, die die Distanz in Metern zwischen zwei geographischen Koordinaten (Breitengrad und Längengrad) auf der Erdoberfläche berechnet. Die Funktion nutzt die Haversine-Formel, welche auf Kugelgeometrie basiert und aus der Quelle [16] movabletype in Javascript übernommen wurde, um die kürzeste Entfernung zwischen zwei Punkten auf der Erdoberfläche zu berechnen.

Die Funktion `getDistanceFromLatLonInMeters` hat vier Parameter: `lat1` und `lon1` sind die Breiten- und Längengrade des ersten Punktes, während `lat2` und `lon2` die Breiten- und Längengrade des zweiten Punktes sind, zwischen denen die Distanz berechnet werden soll.

Zu Beginn des Codes werden die Eingabeparameter auf ihre Gültigkeit geprüft und eine Fehlermeldung wird ausgegeben, falls eine der Koordinaten außerhalb des Bereichs von -90 bis 90 für die Breite und -180 bis 180 für die Länge liegt.

Die Funktion berechnet dann die Differenzen der Breiten- und Längengrade sowie den Radius der Erde (R) in Kilometern. Die Differenzen werden dann in Radianen umgewandelt und die Haversine-Formel wird angewendet, um die Entfernung in Ki-

kilometern zu berechnen. Schließlich wird das Ergebnis in Meter umgewandelt und zurückgegeben.

Die Funktion `deg2rad` wird als Hilfsfunktion definiert, um Grad in Radianen umzurechnen.

Die Funktion gibt 0 zurück, falls die beiden Eingabeparameter denselben Wert haben, um zu vermeiden, dass eine sehr kleine Distanz als Ergebnis ausgegeben wird, wenn es sich um denselben Punkt handelt.

Quelle: <https://www.movable-type.co.uk/scripts/latlong.html>

Bestimmung der nächstgelegenen Entfernung

Um die Privatsphäre der Nachbarn zu gewährleisten, benötigen wir eine Funktion, die uns den nächstgelegenen Abstand zu den Nachbarn gibt, ohne den genauen Abstand preiszugeben. Die Funktion heißt `getNearestDistance` und bekommt einen Meterwert als Parameter.

Die Funktion erstellt ein Array mit den Optionen [100, 200, 500, 1000, 5000, 10000, 15000] und setzt die Variable `nearest` auf den ersten Wert im Array.

Dann wird eine Schleife ausgeführt, die durch jedes Element im Array `options` geht und prüft, welches Element am nächsten zum angegebenen Abstand `distance` liegt. Wenn ein Element näher ist als das bisher am nächsten liegende Element, wird `nearest` aktualisiert.

Schließlich wird überprüft, ob `nearest` größer oder gleich 1000 ist, und je nachdem wird der Abstand entweder in Kilometern oder Metern zurückgegeben. Wenn der Abstand größer oder gleich 1000 ist, wird die Einheit `km` hinzugefügt, ansonsten wird `m` hinzugefügt. Wir haben absichtlich keine Switches oder If-Bedingungen benutzt, da wir es jetzt mit der aktuellen Funktion schneller schaffen, die Abstände zu ändern. Wir evaluieren noch, wie das Array mit den Optionen aussieht.

Nachbarschaft von Nutzer bestimmen

Um unseren App-Nutzern ein besseres Verständnis für die Nachbarschaften zu geben, in denen ihre Nachbarn wohnen, zeigen wir bei jedem Benutzer die Nachbarschaft an, in der sie leben. Diese Information wird automatisch in der Datenbank gespeichert, wenn der Nutzer verifiziert wird. Um diese Funktion zu ermöglichen, verwenden wir die

folgende Funktion, die die geografischen Koordinaten des Benutzers verwendet, um die entsprechende Nachbarschaft mithilfe der OpenStreetMap-API abzurufen:

Die Funktion heißt `getOSMSuburbFromCoords` und nimmt zwei Parameter entgegen: `lat` für die geografische Breite und `lon` für die geografische Länge des Benutzers. Diese Funktion gibt eine Promise zurück, die eine Zeichenfolge `Stringn` mit dem Namen der Nachbarschaft des Benutzers enthält. Die Funktion verwendet die Axios-Bibliothek, um eine GET-Anfrage an die OpenStreetMap-API zu senden. Diese Anfrage enthält die geografischen Koordinaten des Benutzers und die gewünschte Zoomstufe 18, um die Nachbarschaft zu finden. Wenn die Anfrage erfolgreich ist, gibt die Funktion den Namen der Nachbarschaft zurück, der aus den Daten extrahiert wird, die von der API zurückgegeben werden. Wenn der Name der Nachbarschaft nicht verfügbar ist, gibt die Funktion den Namen der Stadt oder der Gemeinde zurück, in der sich der Benutzer befindet. Wenn auch diese Informationen nicht verfügbar sind, gibt die Funktion `null` zurück. Wenn bei der Anfrage ein Fehler auftritt, wird eine Fehlermeldung ausgelöst.

Verifizierungscode format überprüfen

Um die Laufzeit bei der Verifizierung von Cloud-Funktionen zu optimieren, ist es sinnvoll, am Anfang der Verifizierung eine Überprüfung durchzuführen, ob der Verifizierungscode das richtige Format hat. Dazu wird die Funktion `verifyVerificationCode` genutzt werden, welche einen `String` als Parameter erwartet und einen `Boolean`-Wert zurückgibt. In der Funktion wird ein regulärer Ausdruck Regex definiert, um sicherzustellen, dass der Verifizierungscode den Anforderungen entspricht. Der Regex lautet `/^[\a-zA-Z0-9]{10}$`, was bedeutet, dass der Code aus genau 10 alphanumerischen Zeichen bestehen muss. Mit der Methode `checkVerificationCode` wird der übergebene Code auf Übereinstimmung mit dem Regex geprüft und das Ergebnis als `Boolean`-Wert zurückgegeben.

Verifizierungscode generator

Die Funktion `generateRandomVerificationCode` erzeugt einen zufälligen Code mit einer Länge von 10 Zeichen, indem sie eine Kombination aus Groß- und Kleinbuchstaben des englischen Alphabets sowie Ziffern von 0 bis 9 verwendet. Dabei wird `Math.random()` zur Generierung einer Zufallszahl zwischen 0 und 1 genutzt und mit der Länge des Zeichenfolgen-Arrays multipliziert, um eine zufällige Position innerhalb des Arrays

auszuwählen. Anschließend wird das ausgewählte Zeichen an das Ergebnis angehängt und dieser Schritt wird für jedes Zeichen wiederholt, bis eine Zeichenkette der Länge 10 generiert wurde.

Diese Funktion wird in der Cloud-Funktion `generateVerificationCode` verwendet, um einen zufälligen Code zu generieren. Es ist wichtig, dass der Code-Generator in der Lage ist, eine ausreichende Anzahl von Codes zu generieren, damit es keine Kollisionen gibt. Da jeder Code zufällig generiert wird und die Funktion eine zufällige Zeichenkette aus 62 möglichen Zeichen erzeugt, ist es äußerst unwahrscheinlich, dass der gleiche Code zweimal generiert wird. Die Anzahl der möglichen Kombinationen beträgt 62^{10} , was ungefähr 8.39×10^{17} Möglichkeiten entspricht. Daher ist die Wahrscheinlichkeit, dass zwei identische Codes generiert werden, vernachlässigbar.

4.5.7 Algolia Search

4.5.8 Algolia SDK

Typesense Search

4.5.9 Typesense SDK

4.6 Email

Aufgrund des erhöhten E-Mail-Verkehrs des Teams nach dem Wettbewerb Linz haCKt musste eine Lösung gefunden werden, um diesen zentral zu verwalten. Aus diesem Grund wurde eine Gmail-Adresse erstellt, nämlich project.locoo@gmail.com.

4.6.1 Wechsel auf Nochba

Als das Team von Locoo auf Nochba wechselte und eine .com- und .at-Domain erwarb, entstand die Idee, eine eigene E-Mail-Adresse zu erstellen. Beim Versuch, einen E-Mail-Server auf Azure selbst zu hosten, wurde schnell klar, dass dies nicht möglich war. Im Allgemeinen gestaltet sich das Hosten eines E-Mail-Servers schwierig, da viele Cloud-Anbieter dies nicht erlauben.

4.6.2 Hosting-Anbieter

Um den Aufwand zu minimieren, wurde der Hosting-Anbieter Namecheap ausgewählt, der ein sehr günstiges Angebot für ein Jahr E-Mail-Hosting hatte, bestehend aus 2 Monaten gratis und einem Jahr zum Preis von \$8.93.

4.6.3 E-Mail-Adresse und Google-Account

Die Entscheidung fiel auf die E-Mail-Adresse project@nochba.com, die mit dem DNS verbunden wurde. Gleichzeitig wurde ein Google-Account erstellt, um alles einheitlich mit der neuen E-Mail-Adresse zu gestalten.

4.6.4 Zugang und Nutzung

Das gesamte Team hatte Zugang zur E-Mail-Adresse und zum Webmail-Zugriff, obwohl die E-Mail-Adresse hauptsächlich von Martin Hausleitner und Arsham Edalatkhah genutzt wurde.

4.6.5 Vorteile

Im Nachhinein stellte sich die Entscheidung, eine eigene E-Mail-Adresse zu erstellen, als sehr sinnvoll heraus, da dies einen professionellen Eindruck vermittelte und unser Projekt ernster genommen wurde. Es erleichterte uns auch die Verwaltung von E-Mails an einem zentralen Ort, was aufgrund der verschiedenen Anfragen, die im Laufe unseres Projekts eingegangen sind, äußerst praktisch war.

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)

Literaturverzeichnis

- [1] Statista. Worldwide software developer working hours.
- [2] „Flutter,” 2022, Version 2.10. Online verfügbar: <https://flutter.dev/>
- [3] „React Native,” 2022, Version 0.68. Online verfügbar: <https://reactnative.dev/>
- [4] „Apache Cordova,” 2022, Version 11.0.0. Online verfügbar: <https://cordova.apache.org/>
- [5] „Ionic,” 2022, Version 6.0.10. Online verfügbar: <https://ionicframework.com/>
- [6] „Xamarin,” 2022, Version 5.10. Online verfügbar: <https://dotnet.microsoft.com/apps/xamarin>
- [7] M. S. Anwar, „Comparison and evaluation of cross-platform framework and development of a digital health platform using selected framework,” 2021. Online verfügbar: <http://uu.diva-portal.org/smash/get/diva2:1626535/FULLTEXT01.pdf>
- [8] Google. (2021) Firebase. Online verfügbar: <https://firebase.google.com/>
- [9] Firebase. (2023) Security Rules | Firebase. Online verfügbar: <https://firebase.google.com/docs/rules>
- [10] M. Hausleitner. (2023) Firestore Security Rules Nochba. Online verfügbar: <https://github.com/Martin-Hausleitner/Nochba/blob/main/firestore.rules>
- [11] ——. (2023) Storage Security Rules Nochba. Online verfügbar: <https://github.com/Martin-Hausleitner/Nochba/blob/main/storage.rules>
- [12] Google. (2023) Firebase Cloud Functions. Online verfügbar: <https://firebase.google.com/docs/functions>
- [13] F. Team. (2021) Flutter Interactive. Online verfügbar: <https://docs.flutter.dev/development/ui/interactive>
- [14] AppDevCommunity. Flutter State Management Approaches with Examples: setState, BLoC, Rx, Provider, Riverpod. Online verfügbar: <https://medium.com/app-dev-community/flutter-state-management-approaches-with-examples-setstate-bloc-rx-provider-riverpod-aad48f79a255>
- [15] Rasmus Andersson, „Inter Schriftart,” <https://rsms.me/inter/>.
- [16] „Calculate distance, bearing and more between two Latitude/Longitude points,” <https://www.movable-type.co.uk/scripts/latlong.html>, Accessed 2023.

Abbildungsverzeichnis

1	ClickUp Dashboard Kalender	3
2	Statistiken zur plattformübergreifenden Nutzung von Statista [1]	6
3	Vergleich der mobilen Frameworks nach Stand 3.2022 [2, 3, 4, 5, 6]	7
4	Systemarchitektur Diagramm	11
5	Nochba Prototyp in Framer	31
6	Nochba Prototyp in Adobe XD	32
7	Nochba Design System	33
8	Nochba Farbpalette	36
9	App Icons	36
10	Die Schriftfamilie Inter	39
11	Finales Logo	41
12	Logo Design Ideen	41
13	Screenshot des "Name bearbeitenScreens	42
14	Screenshot des "MehrMenüs für Beiträge	42
15	Screenshot des "Beitrag meldenMenüs	42
16	Screenshots verschiedener Versionen des App-Home-Screen	43
17	Screenshot der Nochba Website: nochba.at	44

Tabellenverzeichnis

Quellcodeverzeichnis

1	Security Rules Beispiel	13
2	getDistanceFromLatLonInMeters Funktion	51

Anhang