

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, März 2023

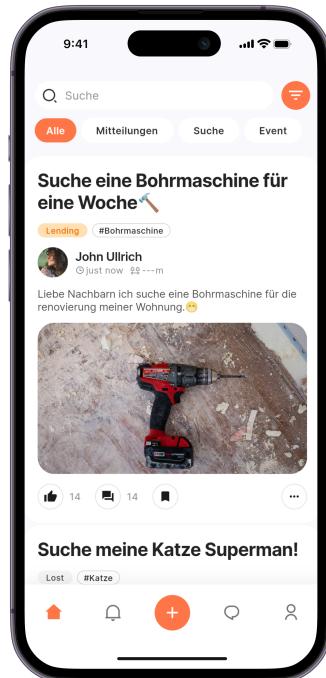
A. Edalatkah & S. Habibovic & M. Hausleitner

Abstract

The 2018 United Nations study [1] shows that the world population will increase to 9.7 billion people by 2050, with the majority of this increase taking place in urban areas. About 68% of the world's population is expected to live in cities by 2050. These developments have far-reaching implications for the way we live and work together in cities.

The Nochba project works to strengthen communities and neighbourhoods in urban areas through various social networking initiatives and measures to promote social connections and cohesion between neighbours. With the increase in urban population by 2050, Project Nochba will play an important role in ensuring that neighbourhoods and communities in cities remain liveable and social isolation is reduced.

The app includes a translation function that helps overcome language barriers between community members. This allows users to search for and offer various forms of help to neighbours, such as shopping for elderly neighbours, helping with renovation work, fixing a flat tyre, looking for a lost pet or borrowing tools. The app also serves as a platform to warn neighbours of possible burglaries or other dangers. Users can verify their address when registering via location recognition or QR codes, with QR codes for communities being passed from phone to phone or published by the city or housing associations. Users can define the size of their community by setting a radius in which they can request or offer help via postings. Neighbours can then see and respond to these postings, and the app facilitates communication between those seeking help and those offering it.

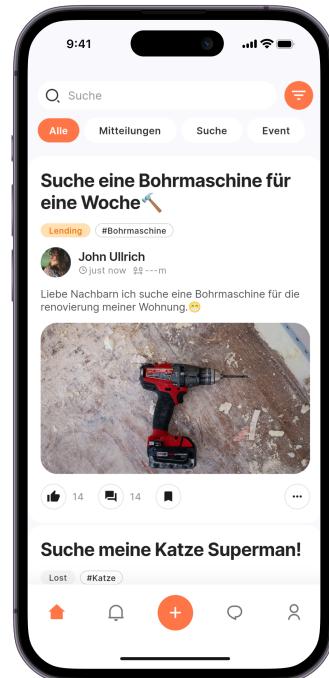


Zusammenfassung

Die Studie der Vereinten Nationen [1] aus dem Jahr 2018 zeigt, dass bis 2050 voraussichtlich ein Anstieg der Weltbevölkerung auf 9,7 Milliarden Menschen erwartet wird, wobei der größte Teil dieses Anstiegs in städtischen Gebieten stattfinden wird. Es wird erwartet, dass bis 2050 etwa 68% der Weltbevölkerung in Städten leben werden. Diese Entwicklungen haben weitreichende Auswirkungen auf die Art und Weise, wie wir in Städten leben und zusammenarbeiten.

Das Projekt Nochba setzt sich dafür ein, die Gemeinschaft und Nachbarschaften in städtischen Gebieten zu stärken, indem es verschiedene Initiativen und Maßnahmen ergreift, um soziale Verbindungen und Zusammenhalt zwischen den Nachbarn zu fördern. Angesichts des Anstiegs der städtischen Bevölkerung bis 2050 wird das Projekt Nochba eine wichtige Rolle dabei spielen, sicherzustellen, dass die Nachbarschaften und Gemeinschaften in Städten weiterhin lebenswert bleiben und dass soziale Isolation reduziert wird.

Die App enthält eine Übersetzungsfunktion, die hilft, Sprachbarrieren zwischen den Mitgliedern der Gemeinschaft zu überwinden. So können die Nutzer unterschiedliche Formen der Nachbarschaftshilfe suchen und anbieten, beispielsweise für ältere Nachbarn einkaufen, bei Renovierungsarbeiten helfen, eine Reifenpanne beheben, ein entlaufenes Haustier suchen oder Werkzeuge ausleihen. Die App dient auch als Plattform, um Nachbarn vor möglichen Einbrüchen oder anderen Gefahren zu warnen. Die Nutzer können ihre Adresse während der Registrierung über die Standorterkennung oder QR-Codes verifizieren, wobei QR-Codes für Gemeinschaften von Telefon zu Telefon weitergegeben oder von der Stadt oder Wohnungsbaugenossenschaften veröffentlicht werden. Die Nutzer können die Größe ihrer Community definieren, indem sie einen Radius festlegen, innerhalb dessen sie über Postings Hilfe anfordern oder anbieten



können. Die Nachbarn können dann diese Beiträge sehen und darauf antworten, und die App erleichtert die Kommunikation zwischen Hilfesuchenden und Helfern.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation	1
1.2 Aufgabenstellung	1
1.3 Projektresultate und wichtige Meilensteine	2
1.4 Projektvorgehensmodell	20
2 Technologien	22
2.1 Technologieevaluierung	22
2.2 Technologieentscheidung	27
3 Systemarchitektur	29
3.1 Flutter	30
3.2 Firebase	31
3.3 Algolia Search	36
4 Umsetzung	38
4.1 Continuous Integration/Delivery	38
4.2 Mobile Anwendung	42
4.3 UI/UX Design	79
4.4 Website	93
4.5 Backend	95
4.6 Verwaltungslogik	118
4.7 Flutter Packages	120
4.8 Email	121
4.9 Visual Studio Code Extention	123
5 Evaluieren getroffener Entscheidungen	124
5.1 Zukünftige mögliche Implementierungen	124
5.2 Erfahrungen	127

Literaturverzeichnis	VII
Abbildungsverzeichnis	XII
Quellcodeverzeichnis	XIV

1 Einleitung

1.1 Motivation

In der heutigen sich schnell entwickelnden Welt verändert die Urbanisierung die Art und Weise, wie die Menschen leben, rapide, was zu einer zunehmenden Anonymität und Unverbundenheit innerhalb der Nachbarschaft führt. Das Team Nnochba hat diese Herausforderung erkannt und eine Möglichkeit gefunden, Gemeinschaften zusammenzubringen und die Verbindung, Zusammenarbeit und Unterstützung zwischen Nachbarn zu fördern.

Durch die Entwicklung einer Social-Media-App, die speziell für die Kommunikation in der Nachbarschaft entwickelt wurde, hofft das Team Nnochba, die Kluft zwischen den Gemeindemitgliedern zu überbrücken und ihnen zu helfen, Sprachbarrieren, kulturelle Unterschiede und die Herausforderungen des modernen Stadtlebens zu überwinden. Das Team möchte die Nachbarschaft, wie man sie in den Dörfern kennt, in die Städte bringen.

1.2 Aufgabenstellung

Das Team Nnochba hat sich auf die Entwicklung einer auf die Nachbarschaft fokussierten Social-Media-App konzentriert, die den Zusammenhalt der Gemeinschaft stärken und den Austausch von Ressourcen unter den Nachbarn erleichtern soll. Das Team hat eine Reihe von Meilensteinen festgelegt, um einen systematischen und effizienten Ablauf des Projekts zu ermöglichen.

Das Team hat sich auf die folgenden Meilensteine geeinigt:

- Entwurf eines Prototypen für die Benutzeroberfläche.
- Festlegung der Systemarchitektur und Überprüfung der technischen Machbarkeit.
- Implementierung eines Minimum Viable Product (MVP) mit Funktionen wie Login, Feed, Beitragserstellung, Filter und Chat-Funktion.

- Ein Minimum Loveable Product (MLP) entwickeln, das Registrierung, Suche, Kontoeinstellungen, Aktivitäten-Tab, Übersetzung, Kommentarfunktion, mehrere Beitragskategorien und ein Reportsystem umfasst.
- Vervollständigung der Implementierung aller geplanten Funktionen.
- Beseitigung von Schwachstellen durch Kundenfeedback und Bugfixing.
- Fertigstellung und Einreichung des Diplomarbeitsprojekts.

Das Team Nochba beabsichtigt, die Arbeit an diesem Projekt auch nach Abschluss der Diplomarbeit fortzusetzen und die App zu verfeinern und zu erweitern, um die Communities besser zu unterstützen und die Verbindung zwischen den Nachbarn zu fördern.

1.3 Projektresultate und wichtige Meilensteine

- Auszeichnungen:
 - #1 Platz Linz hACKT Event X Innovationshauptplatz
 - #1 Platz mPreneur Austria Contest X ICT4D.at
 - #1 Platz Immotopia Innovation Award
 - #3 Platz Spusu Innovation Award (Wien)
 - Siegerteam bei mPreneur School Ohrid X World Summit Award
 - Projektpräsentation für den Bürgermeister von Linz und die Magistratsdirektorin
- Teilnahme an Workshops:
 - Workshop Planet Linz Days
 - Workshop Jugendhackt Österreich
 - Workshop Project-Forge ICT4D.at
- Medienauftritte:
 - UNESCO Internet4trust Konferenz
- Bevorstehende Events:
 - World Summit Award (Graz)

- **Wichtige Meilensteine:**

- **Veröffentlichung von der *Nochba App* im Play Store**

Seit Beginn des Diplomarbeitsprojekts hat sich das Nochba Team auf die Entwicklung einer App mit dem Fokus auf soziale Aspekte im Bereich *Ressourcen-Sharing* und *Community Building* konzentriert. Das Ziel des Teams war es, ein Konzept zu entwerfen, das es ermöglicht, auch nach Abschluss der Diplomarbeit einen wertvollen Beitrag für die Gesellschaft zu leisten. Um die IT-Kenntnisse zu vertiefen und von Mentoren und Branchenexperten mehr über das Thema der sozialen Innovation zu erfahren, entschied sich das Team, an einer Reihe von Wettbewerben und Veranstaltungen teilzunehmen.

Durch diese Erfahrungen konnte das Team Nochba eine solide Grundlage für das Projekt schaffen und war davon überzeugt, dass dies ein fundamentaler Bestandteil der Vision sein wird, die das Team zu erfolgreichen Ergebnissen führen wird. Die Kompetenz des Teams im Bereich der nachhaltigen Innovation und sozialen Unternehmerschaft wurde durch die Teilnahme an diesen Wettbewerben unterstrichen.

- **Linz-hACKT:**

- **11. bis 13. März 2022**

Während des dreitägigen Linz Hackathon-Events [2] hatte das Team Nochba fünf Mentoren aus verschiedenen Bereichen der IT-Industrie in Österreich und Deutschland, mit denen Coaching-Sessions geplant wurden. Diese Mentoring-Sessions halfen dem Team, die Idee nicht nur aus technischer Sicht zu betrachten, sondern auch aus wirtschaftlicher und sozioökonomischer Perspektive zu analysieren. Themen wie Barrierefreiheit und Sicherheit wurden mit erfahrenen Fachleuten aus der Industrie diskutiert, was dem Team einen ersten Überblick über die Vor- und Nachteile des Projekts gab.

Beispielsweise hat das Team gelernt, dass beim Ressourcen-Sharing-Aspekt der App die ausgetauschten Waren beschädigt zurückgebracht oder sogar gestohlen werden können. Die Wahrscheinlichkeit dafür ist jedoch gering, weil die Nachbarn sich untereinander kennen und daher die bereits bestehende soziale Dynamik eine wichtige Rolle spielt. Der Ruf einer Person hängt auch davon ab, in welchem Zustand die ausgeliehene Ware zurückgebracht wird. Die App trägt nicht nur zur Verringerung der Konflikte in der Nachbarschaft bei, sondern fördert auch die Entstehung neuer Freundschaften durch die Ermutigung, anderen Gutes zu tun.

Das Team Nochba hat an dem Wettbewerb Linz hACkT teilgenommen und den ersten Platz erreicht. Dadurch hat das Team die Chance bekommen, gute Beziehungen zur Stadt Linz, zum Innovationshauptplatz Linz und zu Open Common Linz aufzubauen. Am Tag der Preisverleihung erhielt das Team den Linz hACkT-Pokal persönlich von Bürgermeister Klaus Luger.

Seit der Preisverleihung hält das Nochba Team das Innovationshauptplatz-Team ständig über die Fortschritte bei der technischen Weiterentwicklung der App auf dem Laufenden. Einige Monate später hatte das Team die Chance, am 30. Januar 2023 eine Präsentation für Bürgermeister Klaus Luger und die Magistratsdirektorin Ulrike Huemer zu halten. Das Treffen wurde vom Team des Innovationshauptplatzes organisiert und das Ziel war es, einen Zwischenstandsbericht in Form einer Präsentation über die Leistungen während der Diplomarbeit an der HTL Leonding zu liefern.

Die Abbildung 1 zeigt die Preisverleihung des Teams Nochba bei Linz hACkT 2022, bei der der Linzer Bürgermeister Klaus Luger anwesend war.

Folgende Abbildung zeigt das Team Nochba bei der Projektpräsentation des Nnochba-Apps für den Bürgermeister von Linz und die Magistratsdirektorin am 30. Januar 2023.
[Referenz: Abbildung 2]



Abbildung 1: Linz hACkT 2022



Abbildung 2: Projektpäsentation für den Bürgermeister von Linz und die Magistratsdirektorin am 30. Januar 2023

- **ICT4D.at X mPreneur Contest Austria:**

- **22. September 2022**

ICT4D.at [3] ist eine Organisation, die im Jahr 2009 gegründet wurde und sich mit der Förderung und Implementierung von Informations- und Kommunikationstechnologien zur Unterstützung von Menschen in Entwicklungsländern beschäftigt. Die Organisation hat den Fokus auf Innovationen mit den Schwerpunkten Bildung, Gesundheit und Sensibilisierung.

Das Nochba-Team entschied sich auf Empfehlung ihres Betreuungslehrers, am mPreneur Contest Austria teilzunehmen, um ein Netzwerk auf österreichweites Niveau aufzubauen und die Vision des Projekts Nochba gemeinsam mit den mPreneur-Mentoren zu verstetigen. mPreneur - Social Mobile Entrepreneurship worldwide - ist ein durch Erasmus+ gefördertes Programm.

Schließlich hat das Nochba Team es geschafft, eines der beiden Siegerteams in diesem österreichweiten Wettbewerb zu werden. Als Siegerteam präsentierte das Team das Projekt vor einer internationalen Jury aus Rumänien, Philippinen, Tansania, Uganda, Singapur, Kenia und Österreich. Alle Teammitglieder erhielten für ihre Leistung ein

Teilnahmezertifikat von ICT4D und das Nochba Team hat sich dadurch für die Teilnahme an einem interkontinentalen Wettbewerb in Nordmazedonien qualifiziert.

Diese Abbildung 3 zeigt das ICT4D.at X mPreneur Contest Austria Event im Jahr 2022, bei dem das Team Nochba als Gewinner ausgezeichnet wurde.



Abbildung 3: ICT4D.at X mPreneur Contest Austria 2022

- **Planet Linz Days:**
 - **20. bis 23. Oktober 2022**

Der Planet Linz Day [4] ist eine 3-tägige Veranstaltung mit mehreren Workshops, die von den Gewinnern des Linz Hackathon 2022 gehostet werden. Ziel ist es, der Öffentlichkeit die Konzepte nahezubringen, die die Stadtgesellschaft in Linz beeinflussen werden.

Aufgrund des Erfolges bei der Linz hACKT-Veranstaltung wurde das Nochba-Team eingeladen, am Workshop des Planet Linz Days teilzunehmen. Das Team nutzte diese einmalige Gelegenheit, um eine Umfrage in der Stadt Linz durchzuführen, um herauszufinden, wie sich die BewohnerInnen der Stadt eine Social Media Nachbarschaftshilfe-App vorstellen. Nach der Befragung von insgesamt 50 Personen, um die Perspektive der NutzerInnen zu verstehen, stellte das Team fest, dass die folgenden sozialen Dynamiken am wesentlichsten sind:

- **Mitteilung**

- **Frage**
- **Appell**
- **Warnung**
- **Empfehlung**
- **Gefunden**
- **Suche**
- **Hilfe**
- **Verloren**
- **Ausleihen**
- **Event**

Diese Ergebnisse bildeten die Grundlage für die Liste der Kategorien, in denen Beiträge erstellt werden können. Das Team Nochba nutzte diese Ergebnisse und implementierte sie direkt in die Funktion zur Erstellung von Beiträgen in der App, um die App speziell auf die Bedürfnisse der aktuellen Bewohner der Stadt abzustimmen und eine benutzerfreundliche Erfahrung zu schaffen.

- **Jugend hackt Österreich 2022:**

- **28. bis 30. Oktober 2022**

Jugend-Hackt (Quelle: [5]) ist eine Veranstaltung, die von der Non-Profit Organisation *Open Knowledge Austria* organisiert wird. Sie veranstaltet eine Reihe von Hackathons, Events und Workshops. Ziel ist es, Innovation und kreative Teamarbeit unter Jugendlichen im Alter von 12 bis 18 Jahren zu fördern.

Das Team Nochba hat sich für die Teilnahme am Wettbewerb Jugend hackt Österreich 2022 entschieden, weil das Modell der Nochba-App perfekt in das Schema der Veranstaltung passt. Jedes Projekt bekommt die Möglichkeit, seine Projektideen vor den Organisatoren der Veranstaltung zu pitchen. Der Wettbewerb wird aufgezeichnet und kann auf der offiziellen *Dorf TV*-Website und dem offiziellen YouTube-Kanal von Youth Hacks angesehen werden. (Quelle: [6])



Abbildung 4: Jugend hackt Österreich 2022

- **mPreneur School (Ohrid, Nord Mazedonien) X WSA:**
 - **04. bis 09. November 2022**

Während der Woche, in der Arsham Edalatkhanh an der mPreneur School in Ohrid (Quelle: [7]) teilnahm, wurde er mit dem fortgeschrittenen mPreneur School Programm in Form einer non-formalen Ausbildung betreut. Non-formale Ausbildung umfasst Lernaktivitäten außerhalb des formalen Bildungssystems. Zusammen mit den 16 anderen Teilnehmern aus Asien, Afrika und Europa wurde er von Experten aus verschiedenen Bereichen der Industrie unterrichtet. Die Inhalte der 4 Workshops und der 7 interaktiven Sessions waren Nachhaltiges Entrepreneurship, E-Ethik und soziales Bewusstsein, Künstliche Intelligenz, Big Data und Open Data sowie Virtual Reality und Augmented Reality.

Arsham Edalatkhanh ist in dieser Abbildung zu sehen, während er für die 19 World Summit Award Jury Members präsentiert. [Referenz: Abbildung 5]



Abbildung 5: mPreneur School (Ohrid, Nord Mazedonien) X WSA 2022

WSA (World Summit Award) ist eine Initiative, die im Jahr 2003 von der österreichischen Regierung gegründet wurde. Ziel ist es, mit der Förderung von nachhaltigen digitalen Produkten einen Beitrag zur Erfüllung der 17 UN SDGs (Sustainable Development Goals - United Nations) zu leisten. Die 17 Ziele für nachhaltige Entwicklung [8] sollen bis zum Jahr 2030 erreicht werden.

Das Bild in Abbildung 6 zeigt die 17 Ziele für nachhaltige Entwicklung der Vereinten Nationen.



Abbildung 6: 17 Sustainable Development Goals - United Nations (Quelle: [8])

Alle Länder der Vereinten Nationen haben sich auf diese Agenda für die Zukunft geeinigt. Projekt Nochba ist laut der offiziellen mPreneur Webseite (Quelle [9]) in 5 der 17 Sustainable Development Goals kategorisiert.

Hier sind die 5 Kategorien:

- Nachhaltiges Entwicklungsziel Nummer 3:
 - Gute Gesundheit und Wohlbefinden

SDG Nummer 3 ist das wichtigste Ziel, und Projekt Nochba ist nun offiziell eines der Projekte, die helfen, Ziel Nummer drei zu erreichen, nämlich Gesundheit und Wohlbefinden für alle Menschen, unabhängig von ihrem Alter.

Studien der Forscher der Brigham Young University zum Thema *Loneliness and Social Isolation as Risk Factors for Mortality: A Meta-Analytic Review* (Quelle [10]) zeigen, dass das Gefühl der Einsamkeit dem Rauchen von 15 Zigarren pro Tag entspricht. Die Nochba App trägt zu diesem Ziel bei, indem sie eine sichere virtuelle soziale Dynamik innerhalb einer lokalen Gemeinschaft schafft, in der Menschen unabhängig von ihrem kulturellen Hintergrund und der Sprachbarriere Verbindungen aufbauen oder sogar neue Freundschaften schließen können. So wird Projekt Nochba mit der Social-Media-App für die Nachbarschaft genau dieses Problem der Gesundheit und Einsamkeit aufgreifen.

- **Nachhaltiges Entwicklungsziel Nr. 4:**

- **Qualitativ hochwertige Bildung**

Ziel des SDG Nummer 4 ist es, bis 2030 die Bildungsmöglichkeiten für alle Menschen, unabhängig von ihrem Alter, zu sichern. Vor allem in den Entwicklungsländern, wo es schwierig sein kann, eine qualitativ hochwertige Bildung zu gewährleisten. Ein Aspekt der Qualität der Bildung ist der Zugang zu Nachhilfelehrern.

Die Nochba App trägt zu dieser Herausforderung bei, da sie über eine innovative Funktion verfügt, mit der jeder Nutzer seinen eigenen individuellen Radius festlegen kann, innerhalb dessen er sich mit seinen Nachbarn verbinden kann. Diese Funktion ermöglicht es den Nutzern, nach Menschen mit den gleichen Interessen wie sie zu suchen und lokale Veranstaltungen zu organisieren. Unter Umständen können Lerngruppen gebildet werden, zu denen jeder Lehrer in der Nähe beitragen kann, was eine sinnvolle Hilfestellung für Studenten und Schüler sein wird, die in der Nachbarschaft pädagogische Unterstützung benötigen.

- **Nachhaltiges Entwicklungsziel Nr. 11:**

- **Nachhaltige Städte und Gemeinden**

Nachhaltigkeit in Städten und Gemeinden spielt eine wichtige Rolle in der Agenda 2030, indem leistbarer Wohnraum geschaffen und sichere Lebensbedingungen gewährleistet werden. Ziel ist es, die Umweltverschmutzung zu reduzieren, indem öffentliche Verkehrssysteme zugänglicher gemacht werden, was insbesondere den Bedürftigen zugutekommt. SDG 11 hat direkte Auswirkungen auf das Leben der Armen und Benachteiligten, was zu gesünderen und stärkeren Gemeinschaften beitragen wird.

Die Vision der Nnochba App ist es, einen starken Einfluss auf die sozioökonomische Seite des alltäglichen Lebens in städtischen Gemeinden zu haben. Diese Studie über *die Beziehung zwischen Charakteristika der Nachbarschaft und Einsamkeit bei älteren Erwachsenen in Amsterdam, Niederlande: Eine bevölkerungsbasierte Studie* [11] zeigt, dass das Gefühl, nur eine Nummer zu sein, in Großstädten mit vielen Einwohnern häufiger auftritt als in Gemeinden in ländlichen Gebieten.

Die Nnochba App bringt das Konzept der Nachbarschaft, wie es in ländlichen Gebieten bekannt ist, in die Stadtviertel, um das Gefühl der Einsamkeit und Isolation in größeren Gemeinschaften zu bekämpfen und stattdessen die Nachbarschaft zu stärken, indem sie

die gemeinsame Nutzung von Ressourcen und die Bildung von Gemeinschaften unter den Stadtbewohnern fördert.

- **Nachhaltiges Entwicklungsziel Nr. 12:**

- **Verantwortungsbewusster Konsum und Produktion**

Ziel Nr. 12 der SGD hat das Ziel, die weltweite Nahrungsmittelverschwendungen zu halbieren, indem nachhaltige Methoden in Produktion und Konsum gefördert werden. Die Nochba App ist ein Beispiel für die Sharing Economy in städtischen Gemeinschaften, bei der Ressourcen im Allgemeinen geteilt und somit effizienter genutzt werden können. Dadurch können verschwendete Ressourcen in städtischen Gemeinschaften reduziert werden und gleichzeitig wird die soziale Interaktion und der Zusammenhalt in den Gemeinden gefördert. Die Nochba App leistet somit einen wichtigen Beitrag zur Erreichung des Ziels Nr. 12 der SGD und unterstützt die Schaffung einer nachhaltigen Zukunft.

- **Nachhaltiges Entwicklungsziel Nr. 16:**

- **Frieden, Gerechtigkeit und starke Institutionen**

Ziel Nr. 16 der Sustainable Development Goals (SDGs) hat das Ziel, Gewalt und Diskriminierung zu bekämpfen und wirksame Institutionen auf allen Ebenen aufzubauen, die transparent, rechenschaftspflichtig und inklusiv sind. Eine Studie aus dem Jahr 2021 mit dem Titel "COVID-19 and Seniors Food Access, Health, and Well-being: A Rapid Review" (Quelle: [12]) zeigt, wie Sprachbarrieren und kulturelle Unterschiede zu sozialer Isolation bei älteren Einwanderern in Kanada führen können und wie sich die COVID-19-Pandemie auf dieses Problem ausgewirkt hat.

Die Nochba-App trägt dazu bei, Sprachbarrieren zu überwinden und eine breitere Beteiligung von Menschen unterschiedlicher Herkunft und Sprache zu ermöglichen.

- **Immotopia Innovation Award:**

- **15. Februar 2023**

Der Immotopia Innovation Award [13] ist ein Wettbewerb, der von der Regionalzeitung Oberösterreich Nachrichten und Compact Projektentwicklungs GmbH organisiert wird, um Projekte zu prämieren, die in den folgenden Kategorien einen besonderen Einfluss haben:

- **Nachhaltigkeit**

- **Innovation**
- **Positive Einflussnahme soziale Problemstellungen**
- **Digitalisierung**
- **Integration**

Die Nochba-App gewann in der Kategorie Digitale Unterstützung und erhielt eine Förderung. Die Preisverleihung fand am 15. Februar 2023 in der OÖNachrichten Forum Linz statt.

Das Bild zeigt das Team der Nochba-App gemeinsam mit dem Direktor der HTL Leonding bei der Verleihung des Immotopia Innovation Award 2023 (7).



Abbildung 7: Immotopia Innovation Award 2023

- **Workshop Präsentationstechnik (HTL Leonding):**
 - **16. Februar 2023**

Für den Pitch am Ende des mPreneur Social Mobile Entrepreneurship in Ohrid nahm Arsham Edalatkhan an etwa 6 bis 8 Online-Trainingseinheiten zur Vorbereitung mit drei Jurymitgliedern des World Summit Award teil. Er lernte viel über die Kunst des Pitchings und beschloss daher, eines der WSA-Jurymitglieder einzuladen, einen 120-minütigen Workshop über Präsentationstechniken zu halten.

Der Präsentationstechnik-Workshop wurde von Arsham Edalatkhan organisiert, um den Horizont der SchülerInnen der HTL Leonding zu erweitern und ihnen zu helfen, qualitativ hochwertige Präsentationen in verschiedenen Situationen vorzubereiten. Der Inhalt des Workshops besteht aus dem über 10-jährigen Wissen und der Erfahrung des Gastgebers auf dem Gebiet der Präsentation und des Pitchings.

- **ICT4D.at Project-Forge Workshop 2022 (Wien):**

- **30. November 2022**

Die Project-Forge (Quelle: [14]) ist ein einzigartig strukturierter Workshop, bei dem mehrere Mitglieder und zusätzliche Gäste zusammenkommen, um eine Reihe von verschiedenen ICT4D.at-Projekten aus unterschiedlichen Regionen der Welt zu diskutieren.

Ziel ist es, durch die offene Diskussion der Projekte, den Austausch von Tipps, das Hinterfragen und Kommentieren des Prozesses die Möglichkeit zu schaffen, den bestehenden Ideen einen zusätzlichen Wert zu verleihen und die Visionen der einzelnen Projekte zu schärfen.

Aufgrund des Erfolges der Nochba App beim vergangenen ICT4D.at Wettbewerb wurde das Nochba Team auch zu dieser Veranstaltung eingeladen. Das Projekt Nochba wird im Jahr 2023 ein Gastgeberprojekt für die Diskussionen sein.

Das Bild (siehe Abbildung 8) zeigt eine Diskussionsrunde während des ICT4D.at Project-Forge Workshops 2022 in Wien



Abbildung 8: ICT4D.at Project-Forge Workshop 2022 (Wien)

- **UNESCO Internet4Trust Konferenz 2023 (Paris):**

– **23. Februar 2023**

Die von der UNESCO organisierte Internet4Trust-Konferenz (Quelle: [15]) ist eine Veranstaltung zur Förderung von Vertrauen, Sicherheit und Gleichberechtigung im digitalen Raum. Die Konferenz dient als Plattform für den Dialog und die Zusammenarbeit zwischen verschiedenen Stakeholder, darunter Regierungen, Unternehmen des privaten Sektors, Organisationen der Zivilgesellschaft und Lehranstalten.

Das Hauptziel der Internet4Trust-Konferenz besteht darin, die Herausforderungen und Chancen zu diskutieren, die mit dem Aufbau eines vertrauenswürdigen digitalen Umfelds verbunden sind. Dazu gehören Diskussionen über Themen wie Datenschutz, Sicherheit, digitale Kompetenz und die moralische Anwendung von Technologie. Arsham Edalatkahah wurde für die Rolle des WSA Youth Speaker Delegate nominiert, um die Stimme der Jugend während dieser Konferenz zu vertreten, da er am mPreneur Event in Ohrid teilgenommen und sehr gute Ergebnisse erzielt hat. Arsham Edalatkahah nutzte die Gelegenheit auf der UNESCO-Konferenz Internet4Trust 2023 in Paris, um über die vom Team Nochba entwickelte App zu sprechen.

Dies ist ein Bild (siehe Abbildung 9) von Arsham Edalatkhan auf der UNESCO-Bühne zusammen mit dem Moderator Guilherme Canela, Leiter der Abteilung für Meinungsfreiheit und Sicherheit der UNESCO.



Abbildung 9: UNESCO Internet4Trust Konferenz 2023 (Paris):

- **Spusu Innovation Award (Wien):**

- **22. März 2023**

Der Spusu Innovation Award (Quelle: [16]) ist ein bedeutendes Preisverleihungsprogramm, das in Wien organisiert wird. Der Spusu Innovation Award wird von Spusu, einem Mobilfunkbetreiber in Österreich, organisiert und gesponsert.

Das Team Nochba erreichte bei diesem Wettbewerb den dritten Platz in der Kategorie soziale Innovation. Hier ist ein Bild des Teams Nochba zusammen mit der Jury und dem Veranstalter des Spusu Innovation Awards 2023 in Wien. [Referenz: Abbildung 10]



Abbildung 10: Spusu Innovation Award 2023 (Wien)

- **World Summit Award (Graz)**

- **26. bis 28. April 2023**

Der World Summit Award (WSA) in Graz ist ein angesehener internationaler Wettbewerb (Quelle: [17]), der hervorragende digitale Innovationen mit einem starken Fokus auf soziale Aspekte auszeichnet und diese unterstützt.

Das Hauptziel des World Summit Award ist es, digitale Innovationen zu identifizieren und zu fördern, die das Potenzial haben, das Leben der Menschen zu verbessern, sich mit aktuellen gesellschaftlichen Herausforderungen auseinanderzusetzen und zur Erreichung der Ziele für nachhaltige Entwicklung der Vereinten Nationen (SDGs) beizutragen.

Das Projekt Nnochba hat sich erfolgreich für den hoch geschätzten World Summit Award in der Kategorie *Soziale Innovation* qualifiziert. Grund dafür sind die starken sozialen Aspekte der App und ihr bedeutender Beitrag zu den Zielen für nachhaltige Entwicklung der Vereinten Nationen (SDGs).

Die Teilnahme von Project Nnochba am WSA Global Congress wird zweifellos die Sichtbarkeit und den Ruf des Projekts deutlich schärfen und den Weg für eine verstärkte Zusammenarbeit, Investitionen und Unterstützung öffnen, um die Auswirkungen auf die Gemeinschaften weltweit weiter zu verstärken.

- Hochladen von **Nochba** in den Play Store als Early-Access-App

– 22. März 2023

Die Entwicklung einer App umfasst mehrere Phasen, und es muss sichergestellt werden, dass ihre Funktionalität und Benutzerfreundlichkeit den gewünschten Standards entsprechen, bevor sie vollständig veröffentlicht wird. Um dies zu erreichen, hat das Team Nnochba die App *Nochba* als Early Access App zu Testzwecken in den Google Play Store hochgeladen. Auf diese Weise konnten die Nutzer wertvolles Feedback geben und Fehler oder Probleme identifizieren, die behoben werden mussten. Im Folgenden finden Sie einen Überblick über die Schritte, die das Team Nnochba beim Hochladen von *Nochba* in den Play Store als Early-Access-App unternommen hat:

- Vorbereiten der App für die Veröffentlichung

– Das Team Nnochba bereitete die App für die Freigabe vor, indem es sicherstellte, dass sie voll funktionsfähig war und alle erforderlichen Elemente wie Symbole, Screenshots und Materialien für die Präsentation zusammenstellte. Das Team entfernte alle Platzhalter oder Testdaten und bestätigte, dass die App den Richtlinien des Google Play-Entwicklerprogramms und der Vertriebsvereinbarung für Entwickler erfüllt.

- Einrichten der App in der Google Play-Konsole

– In der Google Play-Konsole richtete das Team Nnochba die App ein, indem es die erforderlichen Informationen angab, z. B. den Namen der App, die Standardsprache und den Anwendungstyp, und die entsprechende Preisoption auswählte.

- Vorbereiten des App-Store-Listings

– Das Team Nnochba bereitete die Auflistung der App im Store vor, indem es alle erforderlichen Details bereitstellte, darunter eine Kurzbeschreibung, eine detaillierte Beschreibung, Screenshots, eine Grafik zu den Funktionen und ein App-Symbol, um sicherzustellen, dass die Assets der App den Richtlinien und Spezifikationen von Google Play entsprechen.

- Konfigurieren der Inhaltsbewertung der App

– Das Team Nnochba konfigurierte die Inhaltseinstufung der App, indem es den Fragebogen zur Inhaltseinstufung ausfüllte und so sicherstellte, dass die App der richtigen Zielgruppe angezeigt wurde. In diesem Fall entschied das Team,

dass die Nutzer der App mindestens 14 Jahre alt sein sollten, da solche Apps mit diesem Grad an Komplexität und sozialem Aspekt ein gewisses Maß an Reife erfordern, um verantwortungsvoll genutzt zu werden.

- **Vorbereiten der App für Alpha- oder Beta-Tests**

- Im Bereich *Testen* der Play Console bereitete das Team Nochba die App für Alpha- oder Betatests vor, indem sie eine Testversion erstellten und die APK- oder App-Bundle-Datei der App hochluden. Außerdem fügten sie alle erforderlichen Informationen hinzu, beispielsweise die Versionshinweise.

- **Hinzufügen von Testern und Einladung zum Testen der App**

- Das Team Nochba fügte Tester hinzu und übermittelte ihnen eine Einladung zum Testen der App, indem es die Testmethode (geschlossener oder offener Test) auswählte und den Testern entweder ihre E-Mail-Adressen oder eine URL für den offenen Test zur Verfügung stellte. Das Team informierte die Tester darüber, dass sie Nochba als Early-Access-App über den angegebenen Link aus dem Play Store herunterladen konnten.

- **Überprüfen und Veröffentlichen der App**

- Schließlich überprüfte und veröffentlichte das Team Nochba die App. Nachdem sie alle Informationen und Assets überprüft hatten, klickte das Team auf *Überprüfen und veröffentlichen*, um die App zur Überprüfung einzureichen. Der Überprüfungsprozess nahm mehrere Tage in Anspruch. Danach wurde die App im Play Store als Early Access-App zu Testzwecken zur Verfügung gestellt.

Mit diesen Schritten hat das Team Nochba erfolgreich eine Early-Access-App in den Play Store hochgeladen, um wertvolles Feedback von Testern zu sammeln und die Performance und Benutzerfreundlichkeit der App vor dem eigentlichen Start zu verfeinern.

In folgender Abbildung [Referenz: Abbildung 11] ist ein Screenshot der Nochba-App im Google Play Store zu sehen.

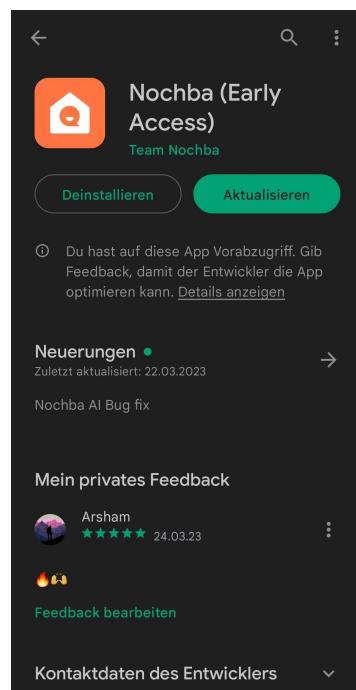


Abbildung 11: Screenshot der Nochba-App im Google Play Store

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 den anderen beiden Teammitgliedern bestätigt wird.

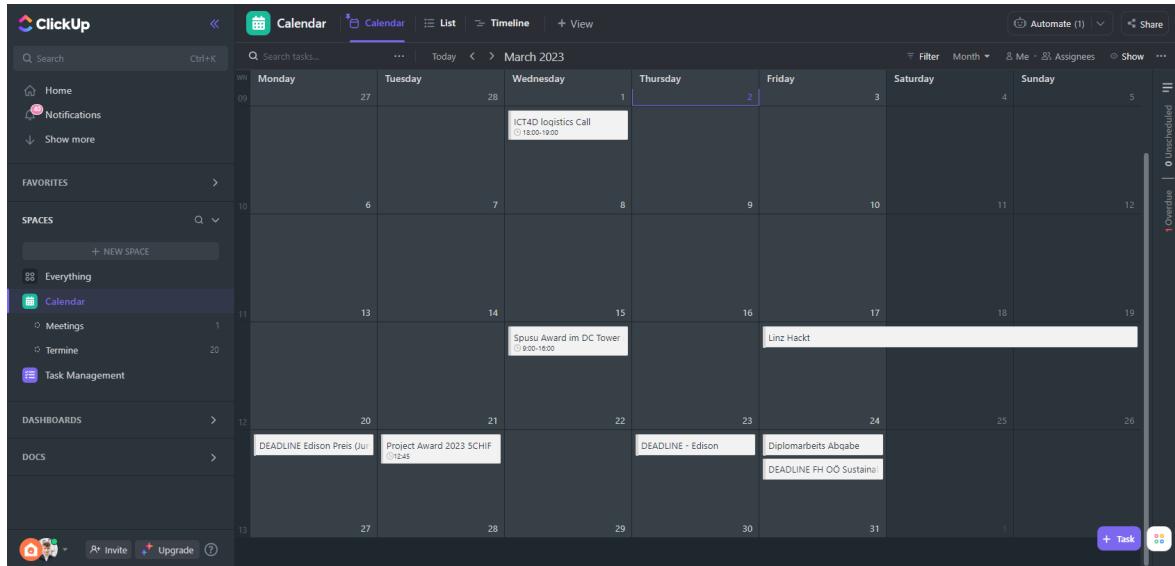


Abbildung 12: ClickUp Dashboard Kalender

In der Diplomarbeit wurden viele Mentoring-Sessions, Meetings mit dem Diplomarbeitbetreuer 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 12 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 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.

Als Resultat einer überlegten Entscheidung wurde festgelegt, dass die Bewältigung dieser Aufgabe nicht durch native Programmierung erfolgen soll. Die mangelnde Vertrautheit aller Teammitglieder mit den Plattformen würde zu einem doppelten Arbeitsaufwand führen, der für ein kleines Team von lediglich drei Personen unverhältnismäßig wäre.

Um eine effiziente und plattformübergreifende Entwicklung sicherzustellen, fiel die Wahl auf die Verwendung eines Cross-Platform-Frameworks. Dabei wurde die Statistik in Abbildung 13 herangezogen, um die am häufigsten genutzten Plattformen zu ermitteln. Es stellte sich heraus, dass Flutter mit 42 Prozent das am meisten verwendete Framework ist und die Tendenz steigend ist. React Native belegt mit nahezu gleicher Beliebtheit den zweiten Platz.

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.

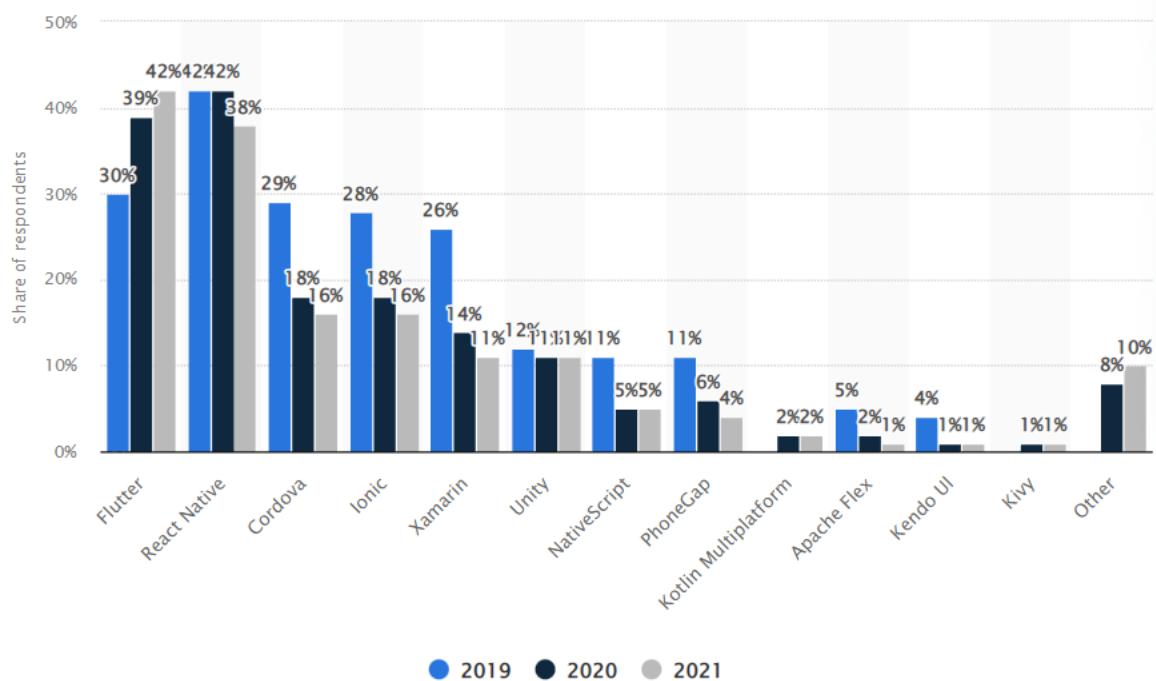


Abbildung 13: Statistiken zur plattformübergreifenden Nutzung von Statista [18]

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 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 14: Vergleich der mobilen Frameworks nach Stand 3.2022 [19, 20, 21, 22, 23]

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 dem 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 jedes 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 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 weist er 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 [24] wurden alle genannten Plattformen bis auf Cordova verglichen, wobei ein aussagekräftiger Vergleich 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 der Frameworks Flutter bis zum aktuellen Zeitpunkt erheblich verbessert.

2.1.2 Backend

Durch umfassende Kenntnisse des Teams in verschiedenen Backend-Technologien wurde die Entscheidungsfindung für das Projekt erleichtert. Bei der Auswahl lag der Fokus auf 'Backend-as-a-Service'-Plattformen, da diese bereits viele Funktionen implementiert haben und das Team dadurch nicht alles von Grund auf programmieren musste.

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.

Somit verblieben Appwrite und Firebase als die beiden vielversprechendsten Optionen. Obwohl Appwrite in Bezug auf die Funktionen nicht mit Firebase mithalten konnte, spricht für Appwrite, dass es zu 100 Prozent Open Source ist, während Firebase Closed Source ist. Da noch keine endgültige Entscheidung bezüglich des Cross-Platform-Frameworks getroffen wurde, fiel es nun leichter, sich für Firebase zu entscheiden, da es das beste Flutter-SDK aufwies, was an den verfügbaren Features erkennbar war.

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

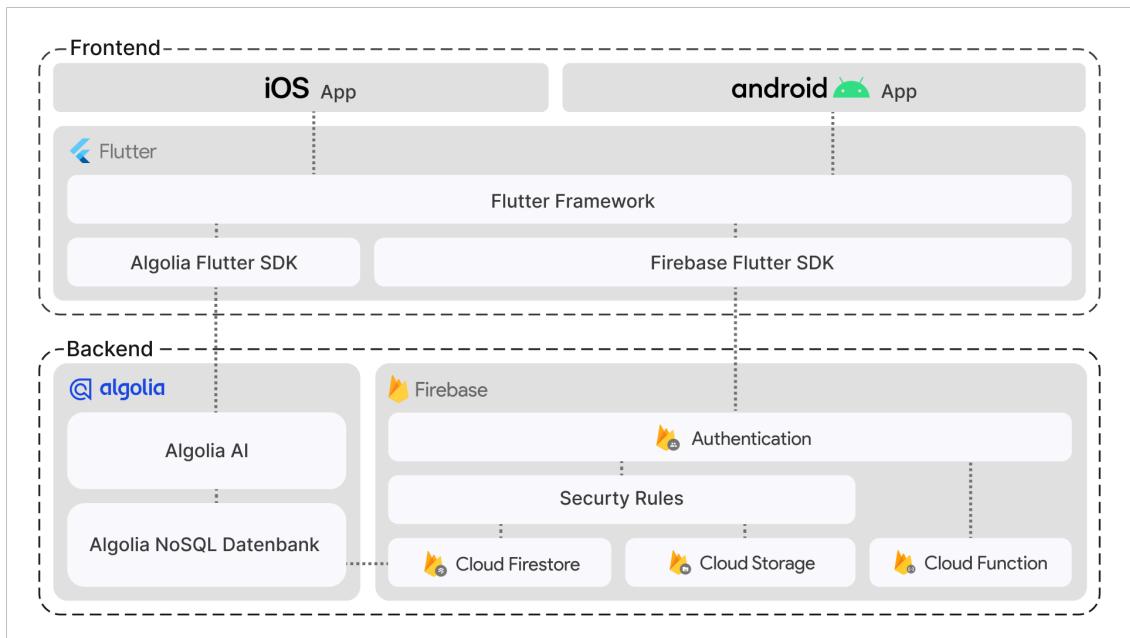


Abbildung 15: Systemarchitektur Diagramm

Die Abbildung 15 veranschaulicht vereinfacht die 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 Packages Algolia Flutter SDK und Firebase Flutter SDK genutzt. Zur Vereinfachung wurden die Funktionen der Firebase Flutter SDK in einem einzigen Block 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.

Die Nutzung von Firestore, einer NoSQL-Datenbank von Firebase, ermöglicht die Speicherung von Daten. Um auf die Firestore-Datenbank und den Cloud Storage zuzugreifen, ist es erforderlich, die Firestore Security Rules zu durchlaufen. Diese Regeln steuern sämtliche Zugriffsrechte auf Dokumente in der Datenbank oder dem Cloud Storage. Weiterführende Informationen dazu sind unter dem Abschnitt Security Rules zu finden.

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

Flutter[19] ist ein crossplatform Frontend-Entwicklungsframework von Google, das die Entwicklung von hochperformanten, nativen Apps für iOS und Android vereinfacht. Flutter verwendet eine Widget-basierte Architektur[25], bei der jedes Element der Benutzeroberfläche als Widget definiert wird. Diese Architektur ermöglicht es, schnell benutzerdefinierte Benutzeroberflächen zu erstellen, die auf allen Plattformen konsistent und schnell laufen.

Ein weiterer Vorteil von Flutter ist, dass es eine hohe Entwicklerproduktivität ermöglicht. Flutter bietet eine schnelle Iteration und ein Hot-Reload-Feature, das es ermöglicht, Änderungen im Code sofort zu sehen, ohne die App neu starten zu müssen. Dies ermöglicht es, schnell Fehler zu beheben und neue Funktionen hinzuzufügen.

Flutter verfügt über eine umfangreiche Sammlung von Bibliotheken und Tools, die auf der offiziellen Paket-Repository-Website von Flutter, pub.dev[26], zu finden sind. Diese Website ist ein Verzeichnis von Paketen, dass von der Flutter-Community erstellt und gewartet wird, und bietet eine Vielzahl von Flutter-Paketen unter Open-Source-Lizenzen an. Dort können nützliche Bibliotheken gesucht und in Flutter-Projekte integriert werden.

Das Vorhandensein so einer umfangreichen Sammlung von Tools und Bibliotheken ist ein wichtiger Faktor, der die Attraktivität von Flutter als Entwicklungsframework erhöht.

3.2 Firebase

Firebase[27] 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 SDK

Firebase SDK[28] 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
- 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.

3.2.2 Firebase Authentication

Firebase Authentication[29] ist eine Technologie, welche die Möglichkeit bietet, Authentifizierung und Autorisierung für Apps und Webseiten unkompliziert und sicher einzubinden. Als Teil der Firebase-Plattform stellt Firebase Authentication eine vollständig gehostete Backend-Authentifizierungslösung bereit, die es erlaubt, sich auf Anwendungsentwicklung zu konzentrieren, ohne sich mit Authentifizierungs- und Sicherheitsimplementierungen auseinandersetzen zu müssen.

Es stehen verschiedene Authentifizierungsoptionen zur Verfügung, wie zum Beispiel:

- E-Mail und Passwort
- Telefonnummer
- Soziale Medien wie Google, Facebook und Twitter
- benutzerdefinierte Authentifizierungsmethoden

Firebase Authentication unterstützt zudem Multi-Faktor-Authentifizierung, um die Sicherheit zu verstärken und den Schutz von Benutzerkonten sicherzustellen.

Die Technologie setzt auf Token-basierte Authentifizierung, um Benutzer zu erkennen und deren Authentifizierungsdaten zu verwalten. Nach erfolgreicher Authentifizierung eines Users kann die App den von Firebase Authentication bereitgestellten Token nutzen, um auf weitere Firebase-Services zuzugreifen. Dazu zählen Firestore für Datenbanken, Storage für Dateien und Functions für serverlose Berechnungen.

Firebase Authentication ist auf verschiedenen Plattformen wie Web, iOS, Android und Flutter verfügbar. Die Integration in Firebase-basierte Apps ist einfach und auch die Nutzung in Verbindung mit anderen Plattformen oder Anwendungen, die nicht auf Firebase aufbauen, ist möglich.

Insgesamt bietet Firebase Authentication eine sichere, skalierbare und benutzerfreundliche Lösung für die Implementierung von Authentifizierung und Autorisierung in Anwendungen und Websites.

Security Rules

In der 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[30][31][32], 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.

Zusammenfassend stellen die Security Rules für Cloud Firestore und Cloud Storage eine wirkungsvolle und anpassungsfähige Option dar, um Zugriffsrechte auf Daten und Ressourcen innerhalb einer Firebase-App zu kontrollieren. Mithilfe der Security Rules lässt sich gewährleisten, dass Nutzerdaten geschützt sind und ausschließlich von berechtigten Personen eingesehen oder verändert werden können.

3.2.3 Cloud Firestore

Cloud Firestore[33] ist eine skalierbare, serverlose NoSQL-Datenbank, die als Teil der Firebase-Plattform angeboten wird. Diese ermöglicht es, Daten in der Cloud zu speichern, Echtzeit-Synchronisierung zwischen verschiedenen Geräten bereitzustellen und Anwendungen ohne die Notwendigkeit einer eigenen Serverinfrastruktur zu erstellen. Dadurch eignet sich Firestore besonders für mobile und Webanwendungen, die auf Skalierbarkeit, Echtzeitaktualisierungen, und hohe Verfügbarkeit angewiesen sind.

Vorteile von Cloud Firestore:

Skalierbarkeit: Firestore ist in der Lage, automatisch zu skalieren, um Millionen von gleichzeitigen Verbindungen und große Mengen von Daten zu unterstützen. Das bedeutet, dass sich um Kapazitätsgrenzen keine Gedanken mehr gemacht werden muss und Anwendungen problemlos erweitert werden können, um einem wachsenden Benutzerstamm gerecht zu werden.

Echtzeitaktualisierungen: Firestore ermöglicht Echtzeitaktualisierungen, indem Änderungen an den Daten automatisch an alle verbundenen Geräte gepusht werden. Dies ist besonders nützlich für Anwendungen, die ständig aktualisierte Daten anzeigen oder Benutzerinteraktionen in Echtzeit verarbeiten müssen.

Flexible Datenstrukturen: Firestore unterstützt flexible Datenstrukturen, die es ermöglicht, Datenmodelle einfach an sich ändernden Anforderungen anzupassen. Dokumente können verschachtelte Felder, Arrays und komplexe Datenstrukturen wie Maps enthalten, was eine effiziente Organisation und Abfrage von Daten ermöglicht.

Leistungsstarke Abfragen: Firestore bietet leistungsstarke Abfragefunktionen, die komplexe Datenabfragen unterstützen. Daten können sortiert, gefiltert und aggregiert werden, um genau die Informationen abrufen zu können, die benötigt werden, ohne unnötige Daten zu übertragen.

Indexes und Abfrageoptimierung: Um Abfragen in Firestore effizienter zu gestalten, verwendet die Datenbank automatisch erstellte Single-Field-Indexes für alle Felder in den Dokumenten. Für zusätzliche Leistung und Komplexität bei Abfragen können auch benutzerdefinierte zusammengesetzte Indexe erstellt werden, die mehrere Felder kombinieren. Diese ermöglichen es, leistungsstarke und komplexe Abfragen auszuführen, ohne die Leistung der Anwendungen zu beeinträchtigen.

Datenmodell

Das Datenmodell[34] in Firestore ist hierarchisch und besteht aus Dokumenten und Sammlungen, auch Collections genannt. Dokumente sind einzelne Datenobjekte, die Felder mit Werten enthalten. Sammlungen sind Container für Dokumente und können weitere Sammlungen enthalten, sogenannte Subcollections. Dieses hierarchische Datenmodell erlaubt es, Daten effizient zu organisieren und abzufragen.

3.2.4 Cloud Storage

Cloud Storage[35] ist eine Technologie von Firebase, die es ermöglicht, Dateien wie Dokumente, Videos und besonders Bilder, wie Profilbilder, sicher und effizient in der Cloud zu speichern und abzurufen.

Firebase Cloud Storage ist ein leistungsstarkes, sicheres und skalierbares Speichersystem, das auf Google Cloud Storage aufbaut und speziell für die Anforderungen mobiler und Webanwendungen entwickelt wurde. Es ermöglicht, Dateien direkt von den Geräten der User in die Cloud hochzuladen, ohne dass ein eigener Server erforderlich ist. Cloud Storage bietet zudem Funktionen wie Sicherheitsregeln, die den Zugriff auf Dateien auf Benutzerbasis steuern, sowie On-the-fly-Komprimierung und Caching, um die Netzwerkbandbreite zu optimieren und die Leistung zu verbessern.

Die Integration von Cloud Storage in die App ist einfach und effizient. Durch die von Firebase bereitgestellten SDKs wird der nahtlose Zugriff auf Cloud Storage und Dateioperationen wie Hochladen, Herunterladen, Anzeigen und Löschen ermöglicht.

Ein wesentlicher Aspekt von Cloud Storage ist die Sicherheit und Zugriffskontrolle. Firebase Cloud Storage ermöglicht, granulare Sicherheitsregeln zu definieren, die den Zugriff auf Dateien basierend auf verschiedenen Kriterien wie Benutzer-Authentifizierung, Dateipfade und Metadaten steuern. Die Sicherheitsregeln sind in einer einfachen, aber leistungsfähigen Sprache geschrieben und können flexibel an die Anforderungen der App angepasst werden. Durch die Kombination von Cloud Storage-Sicherheitsregeln mit Firebase Authentication kann sichergestellt werden, dass nur berechtigte User auf die entsprechenden Dateien zugreifen und Operationen ausführen können.

Firebase Cloud Storage ist auf Performance und Skalierbarkeit ausgelegt und eignet sich sowohl für kleine als auch für große Anwendungen mit hohem Datenaufkommen. Durch die Verwendung von Google Cloud Storage als Basis profitiert Cloud Storage von der globalen Infrastruktur und den Optimierungen von Google, um eine hohe Verfügbarkeit, geringe Latenz und hohe Übertragungsgeschwindigkeiten zu gewährleisten. Zudem können Funktionen wie automatische Komprimierung und Caching genutzt werden, um die Netzwerkbandbreite zu optimieren und die Ladezeiten für die User zu minimieren. Da Firebase Cloud Storage automatisch skaliert, können Anwendungen bei steigenden Nutzerzahlen und Datenvolumen sicher und reibungslos funktionieren.

3.2.5 Firebase Cloud Functions

Dank Firebase Cloud Functions[36] lässt sich Business-Logik, wie etwa der Registrierungsprozess und viele andere logische Vorgänge, einfach und effektiv in der Cloud umsetzen. Die Entscheidung, die Funktionen in TypeScript zu verfassen, ergibt sich aus zahlreichen Vorteilen, wie statischer Typisierung, verbesserter IDE-Unterstützung und erhöhter Code-Lesbarkeit.

Der Registrierungsprozess verdeutlicht den Einsatz von Firebase Cloud Functions. Statt einer monolithischen Anwendung, welche alles auf einem Server verwaltet, wird die Logik in kleinere Funktionen zerlegt, die jeweils spezifische Aufgaben erfüllen.

Firebase Cloud Functions bieten viele Vorteile, darunter einfache Skalierbarkeit und automatische Verteilung zur Bewältigung hoher Lasten. Die Funktionen lassen sich zudem einfach testen und debuggen, indem lokale Emulatoren genutzt werden, bevor sie in der Cloud bereitgestellt werden.

Insgesamt stellen Firebase Cloud Functions eine ausgezeichnete Möglichkeit dar, Business-Logik in der Cloud abzubilden und Anwendungen zu skalieren und zu optimieren. Durch den Einsatz von TypeScript bleibt der Code sauber, robust und wartungsfreundlich.

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.

Im Zuge der Nochba-App hat sich das Team für die Verwendung der Algolia-Suche in Verbindung mit einem Firebase-Backend entschieden. Aus der Perspektive der Systemarchitektur umfasst dieses Konzept zwei wesentliche Teile: die Firebase-Datenbank zur Speicherung und Verwaltung von Daten und den 'Algolia'-Suchdienst zur Bereitstellung schneller und effizienter Suchfunktionalität.

Die in der Firebase-Datenbank gespeicherten Daten werden mit Algolia synchronisiert, um sicherzustellen, dass die Suchergebnisse aktuell und genau sind. Diese Kombination von Technologien ermöglicht es dem Team, die Echtzeit-Fähigkeiten und die einfache

Skalierbarkeit von Firebase zu nutzen und gleichzeitig von den hoch entwickelten Suchfunktionen von Algolia zu profitieren.

Hinsichtlich der Preisgestaltung bietet Algolia eine kostenlose Stufe an, die 10.000 Suchvorgänge und 10.000 Indexierungsvorgänge pro Monat umfasst. Dieses kostenlose Angebot eignet sich für die Testphase der Nochba-App, weil man damit genügend Such- und Indexierungsoperationen durchführen kann, um die Suche und die Leistung zu analysieren. Wenn die App größer wird und die Anzahl der Nutzer steigt, kann das Team ein Upgrade auf einen kostenpflichtigen Plan in Betracht ziehen, um den höheren Ansprüchen nachzukommen.

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.

4 Umsetzung

4.1 Continuous Integration/Delivery

4.1.1 iOS Build

Das Ziel des Projekts war es, eine [37] 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 Erfahrungen mit Apple hatte, übernahm er die Verantwortung für den Build-Prozess der iOS-App. Zur Erstellung einer [19] Flutter iOS-App ein Mac mit [38] Xcode sowie ein [39] 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 nutzte das Team, 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 [40] 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.

4.1.2 Fastlane

Fastlane (Quelle: [41]) ist ein Open-Source-Tool, das von Entwicklern genutzt wird, um den App-Entwicklungsprozess zu automatisieren. Es bietet eine einfache Möglichkeit, häufig verwendete Aufgaben wie das Erstellen von Builds, das Signieren von APKs, das Hochladen von Builds in App Stores und das Versenden von Builds an Tester zu automatisieren. Fastlane kann in Kombination mit Continuous Integration (CI)-Diensten wie Travis CI, CircleCI und Jenkins verwendet werden, um die Prozesse noch weiter zu automatisieren.

Listing 2: Beispiel zum Einsatz von Fastlane

```

1  default_platform(:android)
2
3
4  desc "Build"
5  lane :build do
6    gradle(
7      task: "assemble"
8    )
9  end
10
11
12 platform :android do
13   desc "Deploy to firebase"
14   lane :deploy do
15     begin
16       firebase_app_distribution(
17         groups: "alpha",
18         release_notes: "Bug fixes and performance improvements",
19         apk_path: "../build/app/outputs/flutter-apk/app-release.apk",
20         firebase_cli_path: "usr/local/bin/firebase",
21         firebase_cli_token: "${secrets.FIREBASE_TOKEN}",
22         app: "${secrets.FIREBASE_APP_ID_ANDROID}",
23       )
24     end
25   end
26 end

```

```
27     desc "Increment build number"
28     lane :IncrementBuildNumber do
29
30         path = '../app/build.gradle'
31         re = /versionCode\s+(\d+)/
32         s = File.read(path)
33         versionCode = s[re, 1].to_i
34         s[re, 1] = (versionCode + 1).to_s
35         f = File.new(path, 'w')
36         f.write(s)
37         f.close
38
39     end
40 end
```

Die erste Zeile definiert die Standard-Plattform für die nachfolgenden Befehle. Es wird festgelegt, dass es sich um die Android-Plattform handelt.

Die nächsten beiden Blöcke definieren die Befehle für das Builden der App und für die Deployment auf Firebase für Android. Der Build-Befehl verwendet Gradle, um die App zusammenzubauen. Der Deploy-Befehl verwendet das Firebase-App-Distribution-Plugin, um die App an eine bestimmte Gruppe von Nutzern zu verteilen. Dafür wird eine .apk-Datei bereitgestellt und es werden Release-Notizen sowie Firebase CLI-Informationen übergeben.

Der letzte Block enthält den Befehl zum Inkrementieren der Build-Nummer für die Android-Plattform. Dabei wird der `versionCode` im `build.gradle`-File der App um 1 erhöht.

4.2 Mobile Anwendung

4.2.1 Dateistruktur

Bei der Entwicklung einer Flutter-App gibt es keine vorgegebene Dateistruktur. Statt dessen können Entwickler ihre eigene Struktur entwerfen und gestalten. Im Folgenden wird beschrieben, wie die Dateistruktur für das betreffende Flutter-Projekt aufgebaut wurde.

Da es keine festen Vorgaben gibt, kann man sich bei der Strukturierung an bewährten Praktiken orientieren. In diesem Fall wurde die Dateistruktur so gewählt, dass sie effizient, übersichtlich und gut organisiert ist, um eine reibungslose Entwicklung und Wartung der App zu ermöglichen.

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.

Rückblickend wäre es möglich, die Dateistruktur anders zu gestalten, beispielsweise durch die Definition des UI als eigenes Package und eine bessere Unterteilung der Pages und Views. Diese Anpassungen könnten zu einer noch übersichtlicheren und besser organisierten Struktur beitragen.

4.2.2 State Management

In Flutter gibt es verschiedene Möglichkeiten[42][43], 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[44] 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 lässt sich eine einheitliche Datenquelle erstellen, auf die alle Komponenten zugreifen können. Dies erleichtert die Wartung und Erweiterung der Anwendung. Zudem bietet GetX eine einfache Möglichkeit, Abhängigkeiten zu verwalten und Zustandsinformationen zwischen verschiedenen Bildschirmen zu teilen. Diese Funktionen tragen zu einer effizienteren und besser strukturierten App-Entwicklung bei.

Insgesamt hat die Verwendung von GetX im Flutter-Framework wesentlich dazu beige tragen, eine effektive und skalierbare Anwendung zu entwickeln, die auf die Bedürfnisse der Benutzer zugeschnitten ist.

GetX verfügt über zwei wichtige Konzepte: GetxController und GetxService

GetxController

GetxController ist eine Klasse, die verwendet wird, um den Zustand eines Widgets oder einer Gruppe von Widgets zu verwalten. Ein GetxController ähnelt dem StatefulWidget von Flutter, bietet jedoch zusätzliche Funktionen wie reaktive Programmierung und Dependency Injection. GetxController werden verwendet, um die Logik eines Widgets von seiner Präsentation zu trennen, was das Warten und Testen vereinfachen soll.

GetxController bieten einige Vorteile. Dazu gehören:

Einfachheit: GetxController sind einfach zu verstehen und zu implementieren und erfordern keine zusätzlichen Boilerplate-Code oder das Erlernen von komplexen Konzepten, um es in einer Anwendung zu verwenden.

Flexibilität: GetxController können in jeder Flutter-Anwendung verwendet werden, unabhängig von ihrer Größe oder Komplexität. Es kann leicht in bestehende Projekte integriert oder für neue Anwendungen verwendet werden.

Kapselung: GetxController kapseln den Zustand und die Logik der Anwendung und ermöglichen eine saubere Trennung von Zuständigkeiten. Dadurch wird der Code besser organisiert und wartbarer.

Listing 3: Beispiel zum Einsatz von einem GetxController in Kombination mit GetView

```

1   import 'package:get/get.dart';
2
3   class HomeController extends GetxController {
4     var counter = 0;
5
6     void incrementCounter() {
7       counter++;
8       update();
9     }
10 }
11
12 class HomePage extends GetView<HomeController> {
13   const HomePage({Key? key}) : super(key: key);
14
15   @override
16   Widget build(BuildContext context) {
17     return Scaffold(
18       appBar: AppBar(
19         title: const Text('Home Page'),
20       ),
21     );
22 }
```

```

21      body: Center(
22        child: Column(
23          mainAxisAlignment: MainAxisAlignment.center,
24          children: [
25            GetBuilder<HomeController>(
26              builder: (_) {
27                return Text(
28                  'Counter: ${controller.counter}',
29                  style: TextStyle(fontSize: 24),
30                );
31              },
32            ),
33            SizedBox(height: 16),
34            ElevatedButton(
35              onPressed: controller.incrementCounter,
36              child: Text('Increment'),
37            ),
38          ],
39        ),
40      ),
41    );
42  }

```

Der vorliegende Code 3 ist ein beispielhafter Code, der demonstrieren soll wie ein GetxController in Kombination mit einem GetView verwendet werden kann, um reaktive Funktionalität in einer Flutter-App zu implementieren.

Der HomeController ist eine Klasse, die von GetxController erbt. Die Klasse definiert eine Eigenschaft *counter* und eine Methode *incrementCounter()*, die den Wert der Eigenschaft um 1 erhöht und *update()* aufruft. *update()* ist eine Methode von GetxController, die dem jeweiligen Widget mitteilt, dass das UI aktualisiert werden soll.

Die HomePage-Klasse erbt von einem GetView-Widget, welches den HomeController implementiert hat. Die GetView-Klasse hat eine eingebaute Variable namens *controller*, die den Zugriff auf den HomeController ermöglicht.

GetxService

GetxService ist eine Klasse, die eine globale Instanz eines Dienstes bereitstellt, die in der gesamten Anwendung verwendet werden kann. GetxService werden in der Regel für Operationen verwendet, die einmal ausgeführt werden müssen, wie zum Beispiel das Erstellen, Updaten oder Löschen von Daten in der Datenbanken, API-Aufrufen und anderen Backend-Services. Im Gegensatz zum GetxController bietet GetxService keine automatische Aktualisierung der UI.

Einige Vorteile von GetxServices sind:

Langlebigkeit: GetxServices bleiben für die gesamte Lebensdauer der App bestehen, was bedeutet, dass sie einmal initialisiert werden und dann für die gesamte Nutzungs-

dauer der App zur Verfügung stehen. Dies ist besonders nützlich für Dienste, die nicht direkt an das UI gebunden sind und keine ständige Aktualisierung benötigen.

Wiederverwendbarkeit: GetxServices können in verschiedenen Teilen der Anwendung wiederverwendet werden, ohne dass sie mehrfach initialisiert werden müssen. Dadurch wird der Code effizienter und einfacher zu verwalten.

Zentrale Verwaltung: GetxServices ermöglichen eine zentrale Verwaltung von Anwendungsdiensten und -ressourcen, wodurch der Code besser organisiert und leichter zu warten ist.

Einfache Integration: GetxServices können leicht in bestehende Flutter-Projekte integriert werden, ohne dass große Änderungen am Code vorgenommen werden müssen. Dies erleichtert die Migration von bestehenden Projekten zu GetX.

Listing 4: Beispiel zum Einsatz von einem GetxService

```

1   import 'package:get/get.dart';
2
3   class MyService extends GetxService {
4       void callService() {
5           Get.snackbar('Service', 'The Service was called');
6       }
7   }
8
9   class HomeController extends GetxController {
10     final MyService myService = Get.find<MyService>();
11
12     void callService() {
13         myService.callService();
14     }
15 }
16
17 class HomePage extends GetView<HomeController> {
18     @override
19     Widget build(BuildContext context) {
20         return Scaffold(
21             appBar: AppBar(
22                 title: const Text('Home Page'),
23             ),
24             body: Center(
25                 child: ElevatedButton(
26                     onPressed: controller.callService,
27                     child: const Text('Call Service'),
28                 ),
29             ),
30         );
31     }
32 }
```

Der vorliegende Code 4 ist ein beispielhafter Code, wie ein GetxService verwendet werden kann, um einen Dienst in einer Flutter-App zu implementieren.

Die MyService-Klasse erbt von GetxService und definiert eine Methode *callService()*. Diese Methode ruft eine Snackbar auf, die dem User eine Meldung anzeigen soll.

Die HomeController-Klasse erbt von GetxController und hat eine Instanz von *MyService* mit Hilfe von *Get.find<MyService>()* initialisiert. Wenn die Methode *callService()*

aufgerufen wird, ruft der Controller die Methode `callService()` der MyService-Instanz auf.

Kriterium zum Einsatz von GetxController und GetxService

Um einen GetxController oder GetxService verwenden zu können, muss der jeweilige GetxController oder der jeweilige GetxService zuerst registriert werden. Dafür stellt GetX sogenannte Bindings zur Verfügung. Diese Bindings erstellen Instanzen von den jeweiligen GetxControllern oder den jeweiligen GetxServices, die dann mit `GetView<Controller>` bzw. `Get.find<Service>()` aufgerufen werden können.

Listing 5: Beispiel von einem Binding

```
1 import 'package:get/get.dart';
2
3 class HomeBinding extends Bindings {
4   @override
5   void dependencies() {
6     Get.put(MyService());
7     Get.put(HomeController());
8   }
9 }
```

4.2.3 Authentifizierung

Bei der Authentifizierung wird zwischen dem Anmelden und dem Registrieren unterschieden.

Anmeldeprozess

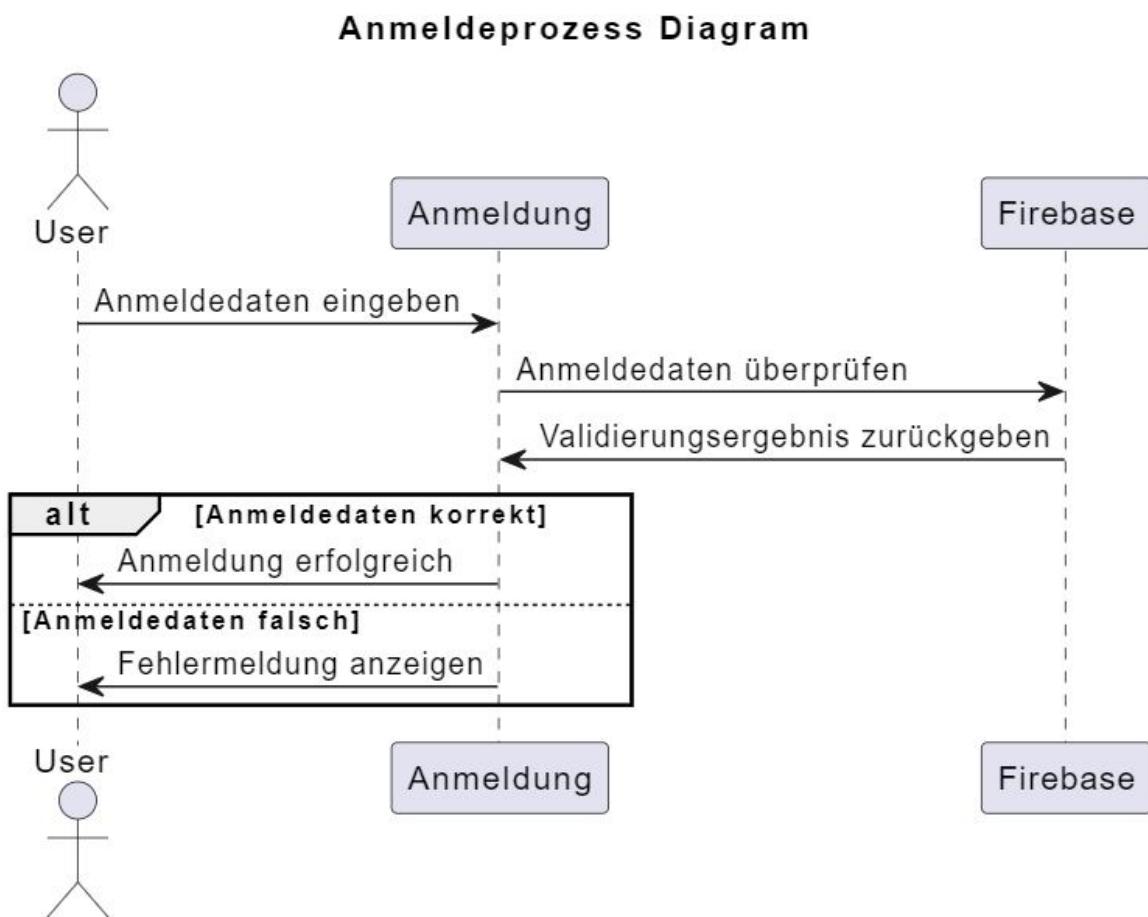


Abbildung 16: Anmeldeprozess

Die Abbildung 16 repräsentiert ein Sequenzdiagramm, welches den Anmeldeprozess für einen User veranschaulichen soll. Der Login-Prozess ist ein wichtiger Schritt in der Benutzerauthentifizierung, bei dem die Identität des Benutzers überprüft wird, um den Zugriff auf die Anwendung zu gewähren oder zu verweigern. Die Anmeldedaten werden an Firebase weitergeleitet, wo diese überprüft werden, um sicherzustellen, dass der User eine gültige E-Mail-Adresse und das korrekte Passwort eingegeben hat.

Falls der User falsche Anmeldedaten eingegeben hat, wird eine Fehlermeldung angezeigt, die den User darauf hinweist, dass die Anmeldedaten falsch sind.

Die Abbildung 17 zeigt, wie die Login-Seite in der App dargestellt wird.

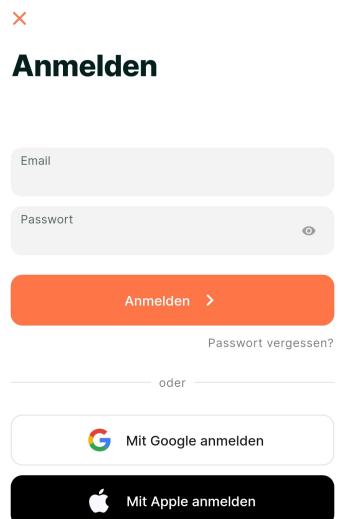


Abbildung 17: Login Ansicht auf der App

Registrierungsprozess

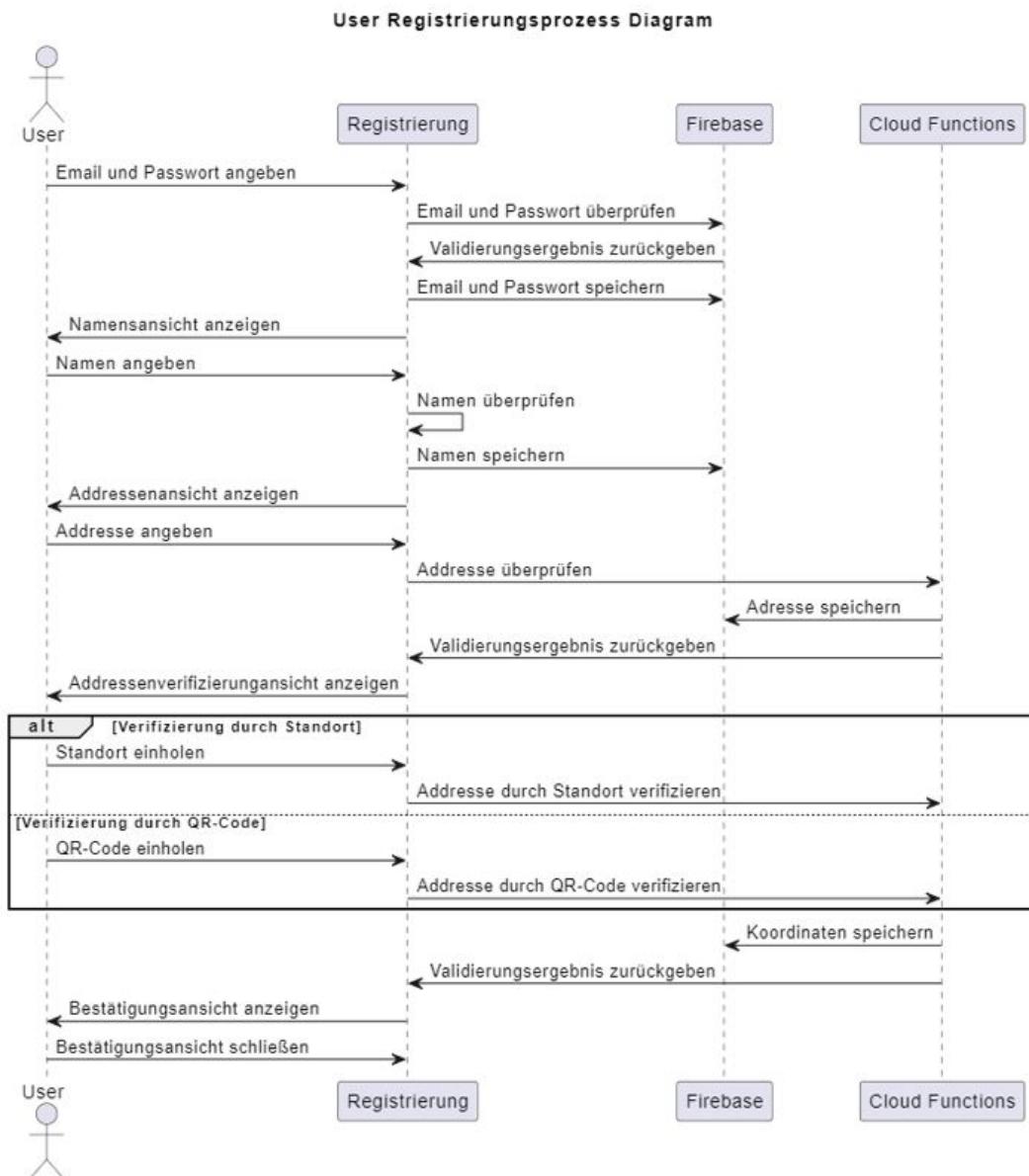


Abbildung 18: Registrierungsprozess

Die Abbildung 18 repräsentiert ein Sequenzdiagramm, welches den Registrierungsprozess für einen User veranschaulichen soll. Der Registrierungsprozess ist ein mehrstufiger Ablauf, wo sichergestellt werden soll, dass alle notwendigen Daten des Users erfasst und validiert werden, bevor die Registrierung abgeschlossen wird. Durch diesen Prozess wird gewährleistet, dass die erfassten Daten korrekt sind und dass nur verifizierte Benutzer Zugriff auf die Anwendung haben.

Der Registrierungsprozess beginnt damit, dass der Benutzer aufgefordert wird, seine E-Mail-Adresse und ein Passwort anzugeben. Das Registrierungsformular leitet dann die E-Mail-Adresse und das Passwort Firebase weiter um es auf dessen Gültigkeit und Richtigkeit zu prüfen. Bei einer erfolgreichen Prüfung werden die Daten eingespeichert

und der User wird eingelogt.

Nach dem ersten Schritt wird der nächste Schritt, die Namensansicht, eingeleitet. Hier wird der Benutzer aufgefordert, seinen Namen und optional schon sein Profilbild anzugeben. Der Name wird dann ebenfalls vom Registrierungsformular auf Korrektheit und Vollständigkeit überprüft und im Firestore eingespeichert.

Nachdem der Benutzer seinen Namen eingegeben hat, wird er zur Addressenansicht weitergeleitet, wo er aufgefordert wird, seine Adresse anzugeben. Wenn der User seine Adresse eingegeben hat, wird sie vom Registrierungsformular zu einer Cloud Function weitergeleitet und validiert. Falls sich die Adresse als eine korrekte Adresse erweist, wird sie direkt von der Cloud Function in den Firestore eingespeichert.

Wenn die Adresse validiert wurde, wird der Benutzer aufgefordert, die Verifizierung seiner Adresse durchzuführen. Die Verifizierung kann durch zwei Methoden erfolgen. Entweder durch eine Überprüfung der Adresse anhand des Standortes oder durch eine Überprüfung der Adresse anhand eines gültigen QR-Codes. Das Registrierungsformular leitet den Standort oder QR-Code an eine Cloud Function weiter, die die Verifizierung übernimmt und bei erfolgreicher Verifizierung, die Koordinaten im Firestore einspeichert.

Wenn die Verifizierung beendet wurde, ist der Registrierungsprozess abgeschlossen.

Während des Registrierungsprozesses wird der User durch eine Authentifizierungsfunktion bereits eingeloggt, da eine Anmeldung für die Ausführung von Cloud Functions erforderlich ist. Wenn der Benutzer den Registrierungsprozess abbricht, werden alle bis zu diesem Punkt gesammelten Daten automatisch gelöscht, um die Datensicherheit und Privatsphäre des Users zu schützen. Wenn der User bei einem Schritt zurückgeht, werden die gespeicherten Daten von diesem Punkt zurückgesetzt, um sicherzustellen, dass nur korrekte Daten im Firestore gespeichert werden und dass keine ungültigen oder unvollständigen Daten gespeichert werden.

Die Abbildung 19 zeigt, wie der Registrierungs-Prozess in der App dargestellt wird.

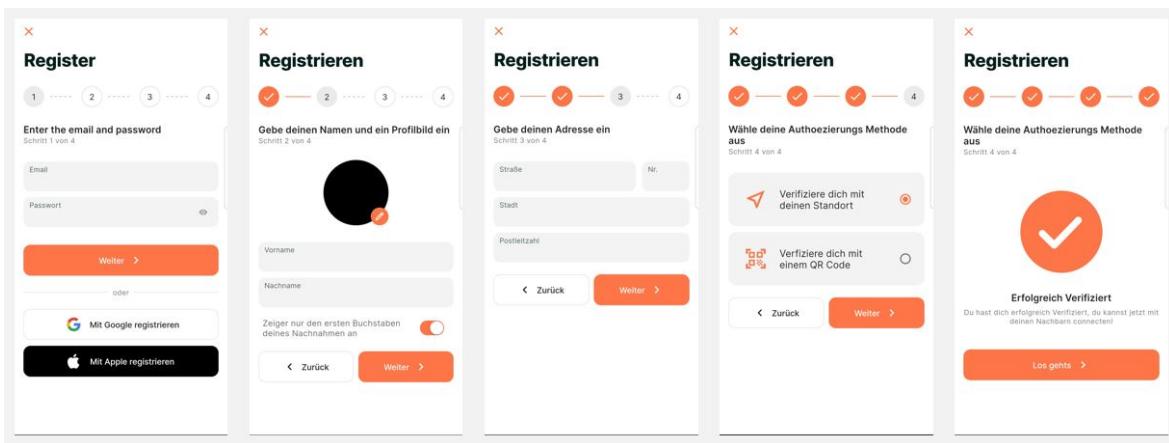


Abbildung 19: Registrierungsprozess in der App

Google Barcode-Scanning ML Kit

In Anbetracht der Entscheidung, Einladungscodes mithilfe von QR-Codes zu implementieren, ist es erforderlich, einen QR-Code-Scanner während des Registrierungsprozesses zu integrieren. Um eine zuverlässige Benutzererfahrung zu gewährleisten, wurde das Google Barcode-Scanning ML Kit als bevorzugte Technologie ausgewählt. In internen Tests zeigte sich, dass dieses Kit eine hohe Genauigkeit und Zuverlässigkeit aufweist. Die Integration des Scanners wurde mithilfe des Pakets [45] `mobile_scanner` realisiert, welches auf der Plattform `pub.dev` verfügbar ist. Dieses Paket ermöglicht eine einfache Implementierung und Anpassung an die individuellen Anforderungen des Projekts. Die Verwendung des ML-Kits trägt dazu bei, die Leistungsfähigkeit der Anwendung zu verbessern und die Benutzerzufriedenheit zu erhöhen.

4.2.4 Feed

Die Feed-Page stellt den zentralen Einstiegspunkt in der App dar und zeigt dem User alle aktuellen Beiträge und Updates aus der Nachbarschaft an.

4.2.5 Beiträge

Beiträge sind das primäre Kommunikationsmittel auf der Plattform und erfordern vor der Veröffentlichung einen Titel, eine Beschreibung sowie eine Reichweite, unter der der Beitrag sichtbar sein soll. Zusätzlich können Bilder und Tags angehängt werden, um den Inhalt des Beitrags besser zu beschreiben und ihn leichter auffindbar zu machen.

Kategorien

Um die Beiträge besser zuordnen zu können, müssen diese vor dem veröffentlichen einer bestimmten Kategorie zugeordnet werden. Diese Kategorien ermöglichen es den Usern, die Art Ihrer Anfrage ihm vorhinein besser zu spezifizieren und die Suche nach bestimmten Arten von Beiträgen zu vereinfachen. Bestimmte Kategorien werden weiters in Unterkategorien aufgeteilt, um die Art des Beitrags nochmals genauer zu spezifizieren. 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 bestimmte Ereignisse oder Nachrichten zu informieren oder sie um eine Meinung oder Empfehlung zu bitten.

Suche: Die Kategorie der Suche ermöglicht es den Nachbarn, im Falle von Hilfebedarf oder einem verlorenen Gegenstand, Kontakt mit ihren Nachbarn aufzunehmen.

Ausleihen: Die Kategorie der 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 bevorstehende Veranstaltung aufmerksam zu machen.

Die Abbildung 20 zeigt, wie die Auswahl der Kategorie bzw. Unterkategorie in der App dargestellt wird.

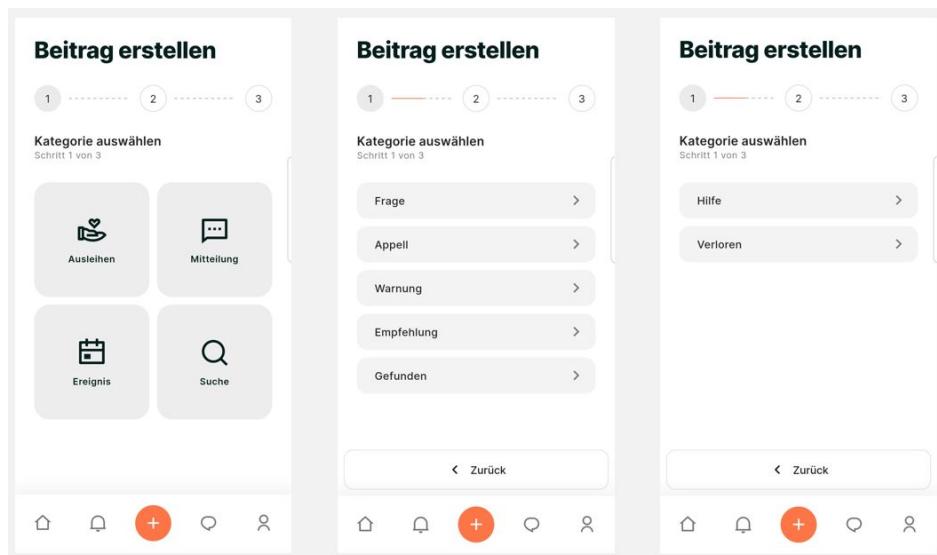


Abbildung 20: Ansicht der Kategorien und Unterkategorien auf der App

Tags

Als Tag wird ein Schlüsselwort beschrieben, dass 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 Informationen über den Beitrag und den Ersteller des Beitrags angegeben werden, wie der Stadtteil, die ungefähre Entfernung zum gegebenen Nachbarn und das Erstelltdatum des Beitrags.

Kommentare

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

Die Abbildung 21 zeigt, wie ein Beitrag mit Tags und die dazugehörige Kommentarsektion auf der App dargestellt werden.

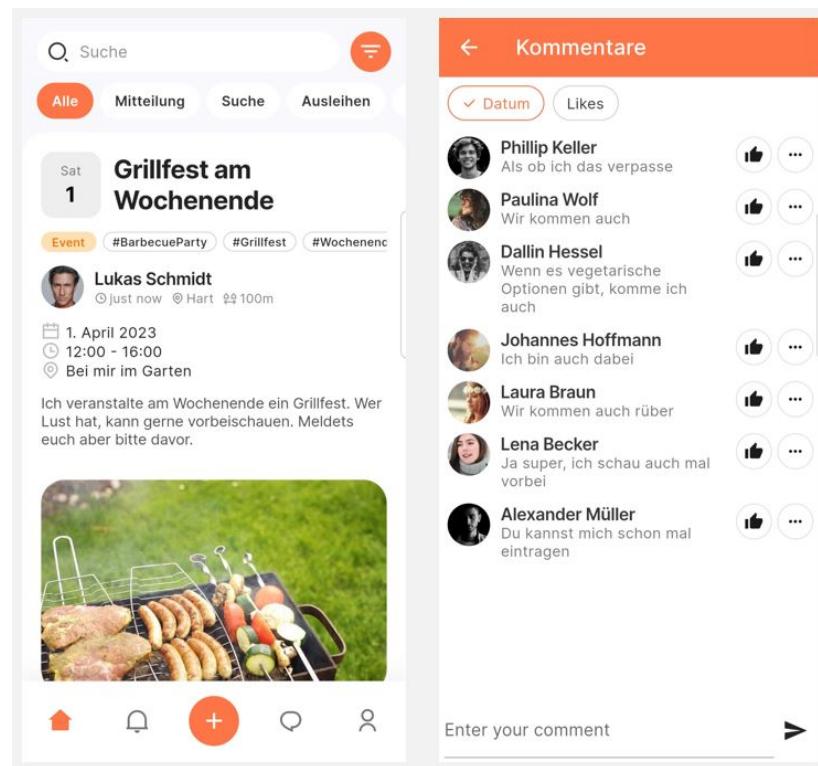


Abbildung 21: Feed-Page und Kommentarsektion auf der App

Beitrag oder Kommentar melden

Um unangebrachte Beiträge oder Kommentare schnell zu identifizieren und auf diese schnell reagieren zu können, gibt es die Möglichkeit Beiträge oder Kommentare zu melden. Diese Meldungen 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

Die Abbildung 22 zeigt, wie das Melden eines Beitrags in der App dargestellt wird.

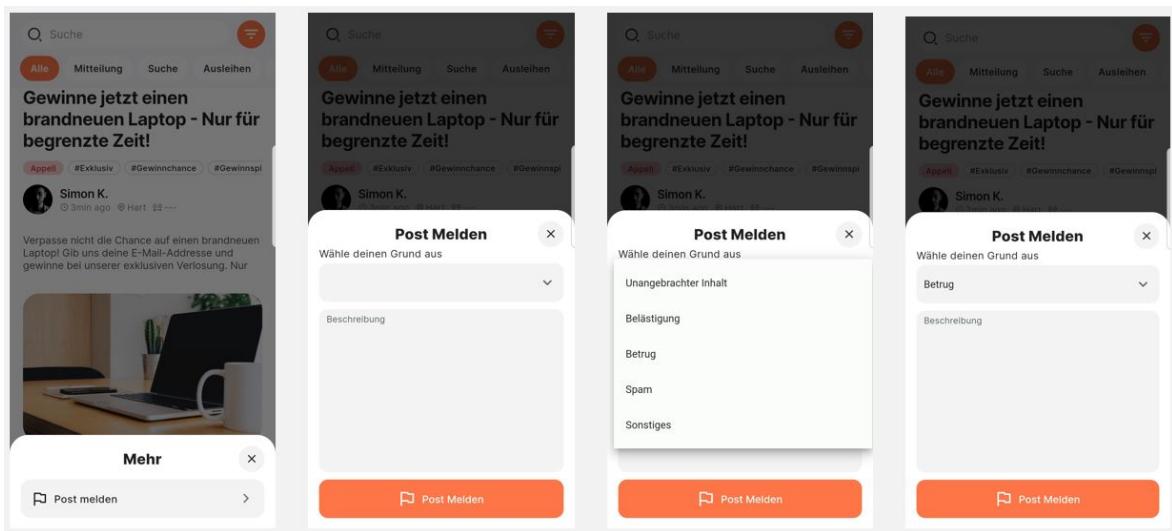


Abbildung 22: Meldeansicht für einen Beitrag auf der App

Autovervollständigung mit künstlicher Intelligenz

GPT-3.5-turbo

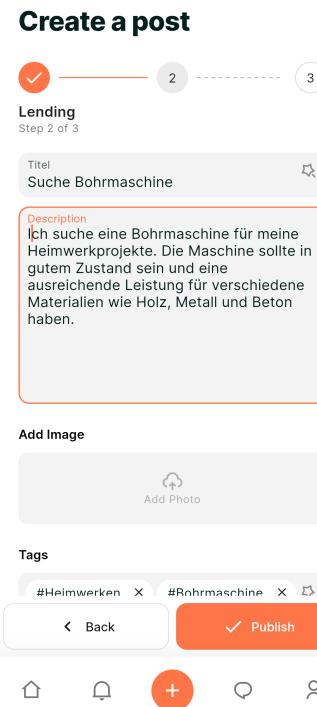


Abbildung 23: Schritt: 2 Beitrag erstellen

In jüngster Vergangenheit haben Fortschritte in der künstlichen Intelligenz (KI) dazu geführt, dass deren Implementierung in verschiedenen Produkten immer einfacher wird. Das Entwicklerteam hat daher beschlossen, auch in Nochba eine vereinfachte

Implementierung durchzuführen. Mit der Veröffentlichung der GPT-3.5-turbo API von OpenAI wurde diese unmittelbar in Nochba integriert. Wie in der Abbildung 23 ersichtlich, befindet sich im Titelfeld ein Zauberstab-Symbol in der rechten oberen Ecke. Bei einem Klick darauf wird eine API-Anfrage an OpenAI gesendet. Um einen Titel basierend auf der Beschreibung zu erstellen, wird folgender Prompt benötigt:

```
beschreibung{ ${widget.descriptionController!.text} } passender Titel:
```

Ein ähnlicher Ansatz wurde auch für die Tags verfolgt. Der Titel und die Beschreibung wurden als Kontext für die KI verwendet. Nach mehreren Versuchen wurde der passende Prompt gefunden, da die KI-Ausgabe zunächst inkonsistent war und nicht erfolgreich in Tags umgewandelt werden konnte. Mit dem folgenden Prompt wurde der Output zuverlässiger:

```
titel{ ${widget.titleController!.text} }
beschreibung --- { ${widget.descriptionController!.text} }
json hashtags richtige sprache: ["#
```

In dem Codeabschnitt 6 sieht man wie die Prompt an die API gesendet wird.

Listing 6: OpenAI GPT-3.5-turbo API Call

```
1 final String prompt = 'titel{ ${widget.titleController!.text} } beschreibung{
2   ${widget.descriptionController!.text} } json hashtags richtige sprache: ["#';
3 final Chat chat = await client.chat.create(
4   model: 'gpt-3.5-turbo',
5   messages: [
6     ChatMessage(
7       role: 'user',
8       content: prompt,
9     ),
10    ],
11  ).data;
```

Die letzten Zeichen ["# wurden absichtlich gewählt, da die KI-Ausgabe manchmal mit oder ohne # geschrieben wurde, was das Parsen der Tags erschwerte. Durch die Klammer und das # werden nun zuverlässig im JSON-Format und Tags mit # erstellt.

Der folgende Code 7 wird verwendet, um die Tags aus der KI-Ausgabe zu parsen und sie anschließend dem TagsController zu übergeben:

Listing 7: parseTags von KI output

```
1
2 List<String> parseTags(String chatOutput) {
3   RegExp regexp = RegExp(r'(?:^|[\n\r\f])#[\w\d]+', unicode: true);
4   Iterable<RegExpMatch> matches = regexp.allMatches(chatOutput);
5   List<String> tags = [];
6
7   for (RegExpMatch match in matches) {
8     String tag = match.group(0);
```

```
9         tags.add(tag);
10    }
11    print(tags.toString());
12    return tags;
13}
14
15}
```

Da die Nutzung der KI kostenintensiv sein kann, wenn mehr Benutzer sie verwenden, ist die Nutzung auf maximal zwei Anfragen pro Benutzer für Titel- oder Tag-Generierung begrenzt. Zukünftig ist geplant, das System durch ein Nochba AI-Abo mit unbegrenzten Anfragen zu monetarisieren.

Übersetzung mit Maschine Learning

Google Translation ML Kit

Für die Übersetzung von Beiträgen vom Deutschen ins Englische wurde eine kosteneffiziente Methode gesucht, um Kosten zu minimieren. Zunächst wurden bekannte Übersetzungs-APIs analysiert, aber dann stieß der Entwickler auf das Google ML Kit, welches 50 Sprachen übersetzen kann. Bemerkenswert ist, dass ein Sprachpaket etwa 5 MB umfasst und auch offline verwendet werden kann. Beim Start der App werden das deutsche und englische Sprachmodell heruntergeladen. Um einen Beitrag zu übersetzen, muss der Benutzer länger auf den Beitrag klicken, um die Übersetzung zu starten.

Google Language identification ML Kit

Um eine Sprache überhaupt übersetzen zu können, benötigt man ein Werkzeug zur Spracherkennung. In dieser App wird das Google Spracherkennungs-ML-Kit verwendet. Bevor der Benutzer einen Beitrag übersetzen möchte, werden die Beschreibung und der Titel an das maschinelle Lernmodell gesendet, um die erkannte Sprache dem Übersetzungs-ML-Kit zu übergeben und eine Übersetzung durchzuführen.

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.

Menüfilter

Die Ansicht vom Menüfilter befindet sich über den Beiträgen und ermöglicht eine schnelle Filterung nach einzig allein den Hauptkategorien.

Die Abbildung 24 zeigt, wie der Menüfilter in der App dargestellt wird.

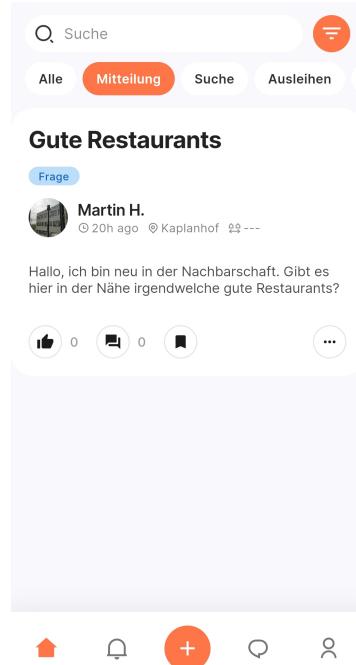


Abbildung 24: Menüfilteransicht auf der App

Hauptfilter

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 Abbildung 25 zeigt, wie der Hauptfilter in der App dargestellt wird.

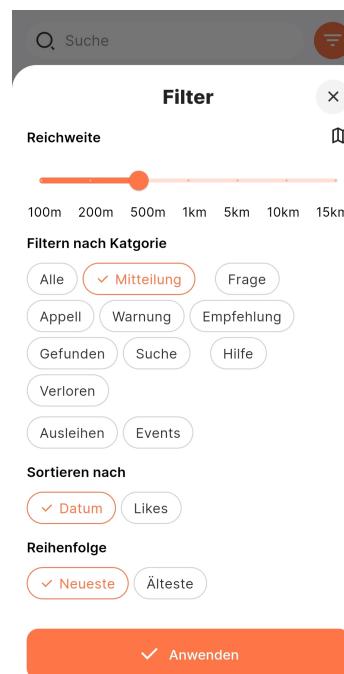


Abbildung 25: Hauptfilteransicht auf der App

Filtern der Reichweite

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.

Die Abbildung 26 zeigt, wie der Reichweitenfilter in der App dargestellt wird.

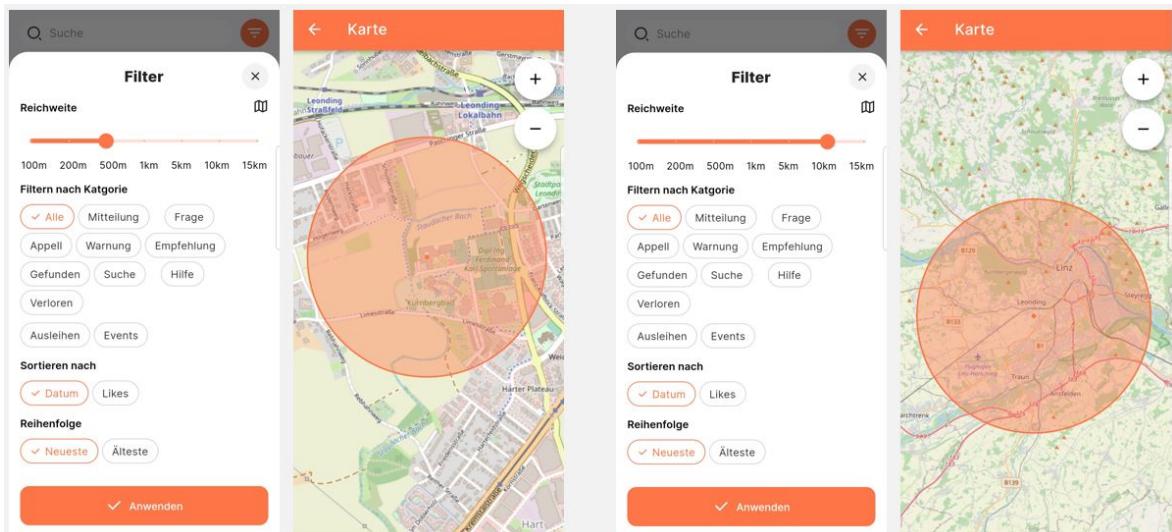


Abbildung 26: Filtern der Reichweite

4.2.7 Suche

Eine gut umgesetzte Suchfunktion in einer Social-Media-App stellt eine wichtige Funktion für den Aufbau einer lokalen Gemeinschaft und die gemeinsame Nutzung von Ressourcen dar. Durch die Verbesserung der Entdeckungsmöglichkeiten und der Personalisierung kann die Suchfunktion erheblich zum Erfolg der Plattform beitragen und die häufige Nutzung durch die Community-Mitglieder fördern.

Eine Suchfunktion kann auf folgende Weise zum Erfolg einer Social Media App beitragen:

- **Entdeckungsmöglichkeit:**

- Eine robuste Suchfunktion ermöglicht es den Nutzern, problemlos wichtige Inhalte, Personen und Ressourcen innerhalb ihrer lokalen Gemeinschaft zu finden. Dies hilft bei der Entdeckung neuer Verbindungen und Veranstaltungen und trägt letztlich zur Verstärkung der lokalen Gemeinschaft und zur Förderung des sozialen Engagements bei.

- **Personalisierung:**

- Die Nutzer können die Suchfunktion nutzen, um Inhalte zu finden, die auf ihre individuellen Interessen und Bedürfnisse zugeschnitten sind. Diese Personalisierung führt zu einer höheren Kundenzufriedenheit, da sich die Nutzer stärker mit der Plattform assoziiert fühlen.

- **Zeiteffizienz:**

- Eine Suchfunktion ermöglicht es den Benutzern, die gesuchten Informationen schnell und effizient zu finden. Das spart Zeit und Mühe, macht die Plattform attraktiver und erhöht die Wahrscheinlichkeit, dass sie häufig genutzt wird.

4.2.8 Typesense

Typesense war aus mehreren Gründen eine ausgezeichnete Wahl für das Nochba-Team:

- **Geschwindigkeit:**

- Typesense ist darauf optimiert, Suchergebnisse schnell zu liefern und so ein flüssiges und reibungsloses Sucherlebnis zu ermöglichen.

- **Tippfehler-Toleranz:**

- Typesense kann mit Tippfehlern umgehen und trotzdem relevante Suchergebnisse liefern. Dies ist eine wichtige Funktion für eine Suchmaschine, da Nutzer beim Tippen oft Fehler machen. Durch die Berücksichtigung dieser Fehler stellt Typesense sicher, dass die Nutzer auch dann korrekte Ergebnisse erhalten, wenn ihre Anfrage Tippfehler enthält.

In diesem Fall entschied sich das Nochba-Team, zu Algolia zu wechseln, das ähnliche Funktionen wie Typesense bietet. Beide Suchmaschinen bieten schnelle, typentolerante und anpassbare Suchergebnisse. Die Entscheidung für Algolia ermöglichte es dem Team, die mit Typesense aufgetretenen Herausforderungen bei der Implementierung zu überwinden und eine effektive Suchfunktionalität innerhalb der App bereitzustellen.

4.2.9 Algolia

Das Team Nochba hat die folgenden Schritte zur Einrichtung und Installation der Algolia-Suche durchgeführt:

- Zusätzlich wurde das Algolia-Paket als *Abhängigkeit* in der Datei pubspec.yaml eingetragen.
- *Flutter packages get* wurde ausgeführt, um die Abhängigkeit zu installieren. Import des Algolia-Pakets in die entsprechende Dart-Datei.
- Initialisierung einer Algolia-Instanz durch Angabe der Applikations-ID und des API-Keys.
- Konfiguration der durchsuchbaren Kategorien durch die Einstellung der *searchableAttributes* im Algolia-Dashboard, die *description*, *title*, *tags* und *category* umfassen.

Testbenutzer für die Entwickler

Das Team Nochba hat beschlossen, auf der Registrierungs-/Anmeldeseite eine weitere Funktion speziell für die Entwickler einzurichten. Diese Option ermöglicht die sofortige Erstellung eines *Testbenutzers*. Nach dem Anklicken der Schaltfläche wird der Testbenutzer automatisch registriert und verifiziert. Der wesentliche Grund für die Entscheidung von Team Nochba war die Vereinfachung des Entwicklungsvorgangs, da sich das Team nicht mehr jedes Mal neu anmelden oder registrieren muss, wenn es die

Applikation während der Entwicklung erneut geladen wird. Dieser Ansatz spart viel Zeit und beschleunigt den Entwicklungsprozess.

Testbenutzer oder Mock-User stehen in der Entwicklungsversion ausschließlich den Developern zur Verfügung, während sie im Release-Modus für die Tester nicht zugänglich sind. Durch die Trennung der Testbenutzer von den tatsächlichen Benutzern stellt das Team Nochba sicher, dass die Entwicklungsumgebung von der Testumgebung getrennt bleibt, um unbeabsichtigte Folgen oder Störungen des Testprozesses zu vermeiden. Diese Strategie ermöglicht es dem Team, eine hohe Softwarequalität beizubehalten und stellt sicher, dass das finale Produkt zuverlässig, sicher und effizient ist.

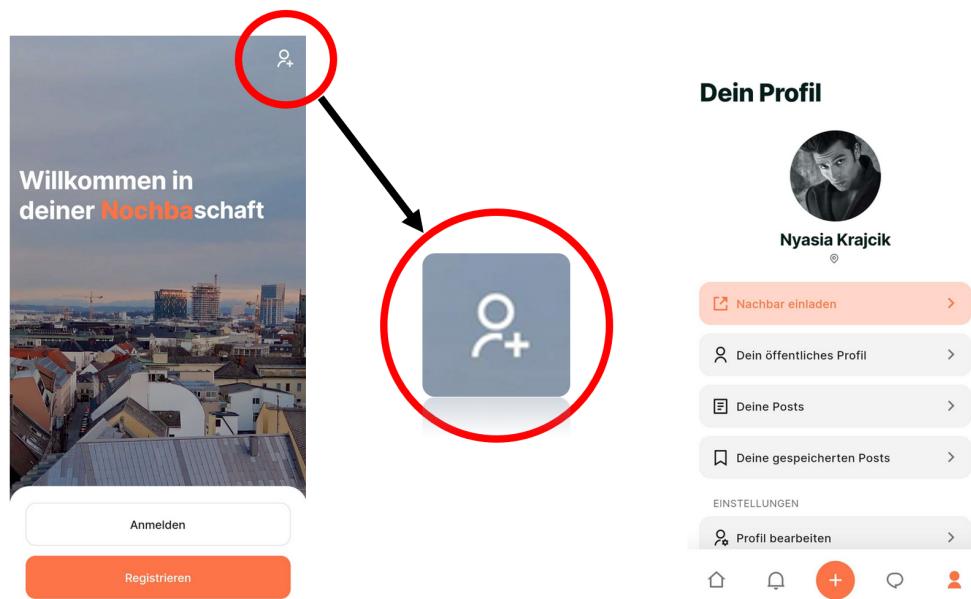


Abbildung 27: Testbenutzer für die Entwickler

4.2.10 Chat

Für die Nutzer einer Nachbarschaftshilfe-App, die sich auf den Aufbau von Gemeinschaften und die gemeinsame Nutzung von Ressourcen spezialisiert hat, bedeutet die Möglichkeit, miteinander zu chatten, dass die Nutzer leicht mit ihren Nachbarn in Kontakt treten, Ideen austauschen, Ressourcen gemeinsam nutzen und stärkere Beziehungen innerhalb ihrer Gemeinschaft aufbauen können. So können sie sich über lokale Geschehnisse auf dem Laufenden halten und haben eine Plattform, um Probleme in der Gemeinschaft zu besprechen und zu lösen.

Die Studie *Core Networks, Social Isolation, and New Media: Internet and Mobile Phone Use, Network Size and Diversity* von Keith Hampton und anderen aus dem Jahr 2011

(Quelle: [46]) untersuchte den Zusammenhang zwischen der Nutzung von Internet und Mobiltelefonen und der Größe und Diversität sozialer Netzwerke. Obwohl diese Studie nicht speziell Nachbarschafts-Apps untersuchte, zeigt sie den potenziellen Nutzen von Online-Netzwerken für den Aufbau von Gemeinschaften und die Stärkung von Beziehungen.

Die Möglichkeit, in einer Nachbarschafts-App wie Nochba zu chatten, kann dazu beitragen, den Austausch von Ideen und Ressourcen unter den Nutzern zu erleichtern, lokale Geschehnisse zu verfolgen und Probleme in der Gemeinschaft zu lösen. Diese Chat-Funktion kann daher dazu beitragen, den Aufbau von Gemeinschaften und sozialem Engagement zu fördern, wie es auch von den Entwicklern von Nnochba betont wird.

Aus Sicht der Entwickler ist die Möglichkeit, mit anderen Nutzern zu chatten, eine wichtige Funktion der App, die den Aufbau von Gemeinschaften und das soziale Engagement fördert. Das Nnochba-Team ist sich bewusst, wie wichtig es ist, starke Community-Verbindungen zu fördern, und ist der Ansicht, dass die Möglichkeit der Kommunikation zwischen den Nutzern eine entscheidende Rolle bei der Erreichung dieses Ziels spielt. Durch die Bereitstellung einer Chat-Funktion ist das Nnochba-Team in der Lage, eine umfassendere und interaktivere Plattform zu schaffen, die die Benutzer dazu ermutigt, sich untereinander auszutauschen und ein stärkeres Gemeinschaftsgefühl zu entwickeln.

Gemäß der Studie *Neighborly Networks: A Study of Online Community Life* von Keith N. Hampton und anderen werden im Folgenden einige weitere Gründe aufgeführt, warum die Möglichkeit, in einer Social-Media-App für die Nachbarschaft miteinander zu chatten, von Vorteil sein kann:

- **Für die Nutzer:**

- Sie können ihre Nachbarn um Empfehlungen für nahe gelegene Dienstleistungen oder Geschäfte bitten.
- Sie können lokale Events oder Treffen organisieren und die Details mit anderen interessierten Mitgliedern der Nachbarschaft besprechen.
- Straßensperrungen, die sich auf die Anwohner auswirken können.
- Sie bietet eine Plattform für den Austausch von Neuigkeiten aus der Umgebung, wie beispielsweise aktuelle Informationen über Bauarbeiten.

- Es kann das Gefühl der Sicherheit fördern, da die Nachbarn im Notfall schnell und einfach miteinander in Kontakt treten können.
- Es kann dazu beitragen, ein stärkeres Gemeinschaftsgefühl zu bilden, da die Nutzer Geschichten, Erinnerungen und Erfahrungen miteinander teilen können.

- **Für Entwickler:**

- Das Engagement der Nutzer und die Zeit, die sie mit der App verbringen, können gesteigert werden, da die Nutzer mehr Zeit mit dem Chatten verbringen können.
- Es kann zu mehr Inhalten führen, die von Nutzern erstellt werden, wie beispielsweise Fotos oder Informationen zu Events, die in der App geteilt werden können.
- Sie kann wichtige Daten über das Verhalten und die Vorlieben der Nutzer liefern, die in künftige Entwicklungentscheidungen einfließen können.
- Es kann die App von der übrigen Konkurrenz abheben und sie für potenzielle Nutzer attraktiver machen.
- Sie kann dazu beitragen, dass Benutzer, die die gemeinschaftsbildenden Aspekte der App zu schätzen wissen, dem Hersteller und seinen Produkten vertrauen und für sie eintreten.

4.2.11 Vergleich von *Flyer Chat* und *Stream Chat Flutter*

Es gibt ein Flutter-Chatpaket namens *Flyer Chat*, das momentan das am besten bewertete Chatpaket auf dem Gebiet ist. Ein weiteres hoch geschätztes Chat-Paket für Flutter-Applikationen ist *Stream Chat Flutter*.

Flyer Chat bietet eine Chat-Benutzeroberfläche und unterstützt die Grundfunktionalität, indem es Entwicklern vorgefertigte Chat-UI-Komponenten wie Chat-Bubbles, Eingabefelder und Nachrichtenlisten zur Verfügung stellt. Dieses Paket ermöglicht es Entwicklern, auf einfache und schnelle Weise Chat-Funktionen in ihre Applikationen zu implementieren und sie an das Design der App anzupassen. Darüber hinaus ist *Flyer Chat* so gestaltet, dass es reibungslos mit gängigen Backend-Diensten für die Speicherung und den Abruf von Chat-Daten zusammenarbeitet und bietet eine über-

sichtliche Dokumentation, Code-Beispiele und Tutorials, um die Benutzerfreundlichkeit zu unterstützen.

Stream Chat Flutter hingegen ist ein SDK, mit dem Entwickler Echtzeit-Chat-Anwendungen für Flutter erstellen können. Es bietet Funktionen wie integrierte UI-Komponenten, Offline-Unterstützung, anpassbare Themen und Unterstützung für mehrere Plattformen, einschließlich Android, iOS und Web.

Für das Team Nochba war es aus folgenden Gründen eine bessere Idee, das *Flyer Chat*-Paket zu verwenden:

- **Integration:**

- *Flyer Chat* bietet eine nahezu reibungslose Installation in bestehende Flutter-Applikationen, was eine unkomplizierte Erfahrung für Entwickler darstellt.

- **Funktionsumfang:**

- *Flyer Chat* verfügt über einen umfassenden Feature-Set, das ein optimales Verhältnis zwischen häufig verwendeten Chat-Funktionen und innovativen, individuellen Eigenschaften ermöglicht.

- **Leistung:**

- Der *Flyer Chat* ist für verschiedene Plattformen optimiert, um ein flüssiges Benutzererlebnis zu ermöglichen, ohne die Leistung der App zu verringern.

- **Dokumentation und Unterstützung:**

- Mit gut dokumentierten Materialien und einer starken Community macht es *Flyer Chat* den Entwicklern besonders einfach, Lösungen für allgemeine Probleme zu finden und Anpassungen vorzunehmen.

4.2.12 Komponentenorientierte Software-Entwicklung

Laut dem 1998 veröffentlichten Fachbuch (Quelle: [47]) *Component-Oriented Programming* von Clemens Szyperski können Komponenten unabhängig voneinander entwickelt, getestet und gewartet werden, was mehrere Vorteile bietet:

- **Wiederverwendbarkeit:**

- Komponenten können über mehrere Applikationen hinweg wiederverwendet werden, was die Entwicklungszeit und den Aufwand verkürzt.

- **Wartbarkeit:**

- Gut definierte Grenzen machen es viel bequemer, Probleme in einer bestimmten Komponente zu erkennen und zu beheben, ohne den Rest der Applikation zu beeinträchtigen.

- **Skalierbarkeit:**

- Anwendungen können durch den Import oder die Substitution von Komponenten ohne größere Anpassungen leicht skaliert werden.

- **Flexibel:**

- Der modulare Aufbau ermöglicht eine einfache Anpassung an sich verändernde Anforderungen.

Die Verwendung des Flutter-Pakets *Flyer Chat* im Kontext der komponentenorientierten Programmierung verdeutlicht die Vorteile dieses Entwicklungskonzepts, insbesondere in der rasanten Welt der heutigen Applikationsentwicklung. Durch die Verwendung eines voll funktionsfähigen Pakets wie *Flyer Chat* können sich Entwickler auf den Aufbau und die Einbindung von Komponenten konzentrieren und so wertvolle Zeit und Ressourcen sparen.

Im Fall von *Flyer Chat* dient das Paket als vorgefertigte Chat-Komponente, die für eine einfache Einbindung und individuelle Anpassung in jede Flutter-Applikation entworfen worden ist. Durch den komponentenorientierten Ansatz bei der Programmierung können Entwickler *Flyer Chat* nutzen, um Chat-Funktionen schnell zu implementieren, ohne eine ganze Chat-Komponente von Grund auf neu schreiben zu müssen. Dies beschleunigt nicht nur den gesamten Entwicklungsprozess, sondern stellt auch sicher, dass die Chat-Funktion nach Standards der Industrie erstellt wird und ein hochwertiges User-Erlebnis bietet.

Darüber hinaus ermöglicht die Verwendung einer vorgefertigten Komponente wie *Flyer Chat* den Developern, mehr Zeit auf andere Aspekte der Applikation zu investieren, wie beispielsweise die Anpassung der Benutzeroberfläche oder das Ergänzen von einzigartigen Funktionen. Dies ermöglicht die Erstellung von benutzerfreundlichen und qualitativ hochwertigen Applikationen, die schneller auf den Markt gebracht werden können.

Zusammenfassend lässt sich sagen, dass die Integration des *Flyer Chat*-Pakets in eine Flutter-Anwendung ein hervorragendes Beispiel für komponentenorientierte Programmierung in der Praxis ist. Dieser Ansatz hilft Entwicklern, Zeit zu sparen, qualitativ

hochwertige Standards einzuhalten und sich auf die Realisierung eines hervorragenden Benutzererlebnisses zu konzentrieren, wovon letztendlich sowohl der Entwickler als auch der Endbenutzer profitieren.

4.2.13 Fotoaufnahme im Chat

Listing 8: Aufnahmeprozess für ein Foto

```

1  void handleTakePhoto(BuildContext context) async {
2      var result = await ImagePicker().pickImage(
3          imageQuality: 70,
4          maxWidth: 1440,
5          source: ImageSource.camera,
6      );
7
8
9      final image = await result!.readAsBytes();
10     if (image == null) {
11         Get.snackbar("null", "null");
12     }
13
14     if (result != null) {
15         setAttachmentUploading(true);
16         final file = File(result.path);
17         final size = file.lengthSync();
18         final bytes = await result.readAsBytes();
19         final image = await decodeImageFromList(bytes);
20         final name = result.name;
21
22     try {
23         String userId = room.users[0].id;
24         final reference = FirebaseStorage.instance.ref('$userId/$name');
25         await reference.putFile(file);
26         final uri = await reference.getDownloadURL();
27
28         final message = types.PartialImage(
29             height: image.height.toDouble(),
30             name: name,
31             size: size,
32             uri: uri,
33             width: image.width.toDouble(),
34         );
35
36         chat.FirebaseAuthCore.instance.sendMessage(message, room.id);
37         setAttachmentUploading(false);
38     } finally {
39         setAttachmentUploading(false);
40     }
41 }
42 }
```

Dieses Codebeispiel definiert eine Funktion, *handleTakePhoto*, die ein Foto mit der Kamera des Geräts über das ImagePicker-Plugin mit einer Bildqualität von 70

Die Funktion *handleTakePhoto* wird definiert, die einen *BuildContext* als Parameter erhält. Diese Funktion wird den Aufnahmeprozess für ein Foto mit der Kamera des Endgeräts regeln.

Innerhalb der Funktion wird die Methode *ImagePicker().pickImage()* mit drei Parametern aufgerufen: *imageQuality*, *maxWidth*, und *source*. Diese Methode gibt ein *Future* zurück, das auf die Datei des jeweiligen Bildes verweist.

imageQuality wird auf 70

maxWidth wird auf 1440 Pixel gesetzt, um sicherzustellen, dass die Bildgröße diese Grenze nicht überschreitet.

source wird auf *ImageSource.camera* gesetzt, was bedeutet, dass das Bild mit der Kamera des Geräts aufgenommen werden soll.

Das Schlüsselwort *await* wird verwendet, um auf die Auflösung von *Future* zu warten, und das Ergebnis wird in der Variablen *result* gespeichert.

Die Methode liest das Bild als Byte-Array mit *await result!.readAsBytes()* und weist es der Variablen *image* zu.

Wenn das Bild null ist, wird mit *Get.snackbar('null', 'null')* eine *Snackbar* mit dem Text *null* angezeigt. Eine Snackbar ist eine kurze, nicht allzu aufdringliche Meldung, die am oberen Rand des Bildschirms angezeigt wird, um dem Benutzer Feedback oder Informationen zu geben.

Wenn das Ergebnis nicht null ist, fährt die Methode mit der Verarbeitung des Bildes fort. Sie setzt einen internen Zustand *setAttachmentUploading(true)*, um anzuzeigen, dass ein Anhang hochgeladen wird. Die Bilddatei wird mit *File(result.path)* initialisiert, und die Dateigröße wird mit *file.lengthSync()* ermittelt. Das Byte-Array des Bildes wird mit *await result.readAsBytes()* gelesen, und das Bild wird mit *await decodeImageFromList(bytes)* entschlüsselt. Der Dateiname des Bildes wird mit *result.name* ermittelt.

Die Methode ladet das Bild in Firebase Storage hoch. Zunächst wird eine Speicherreferenz mit der ID des Benutzers und dem Namen des Bildes mit *FirebaseStorage.instance.ref('userId/roomId')* erstellt. Die Bilddatei wird dann mit *await reference.putFile(file)* hochgeladen. Die Download-URL für das hochgeladene Bild wird mit *await reference.getDownloadURL()* ermittelt.

Es wird ein *types.PartialImage* Objekt mit der Bildgröße, dem Namen, der URL und der Breite des Bildes erstellt.

Das Bild wird als Nachricht mit *chat.FirebaseAuthCore.instance.sendMessage(message, room.id)* gesendet.

Nachdem das Bild erfolgreich hochgeladen und gesendet wurde, wird der interne Status aktualisiert, um anzuzeigen, dass das Hochladen des Anhangs mit *setAttachmentUploading(false)* abgeschlossen ist.

Wenn während des Hochladens des Bildes oder des Versendens der Nachricht eine Fehlermeldung auftritt, sorgt der Final Block dafür, dass der Status mit `setAttachmentUploading(false)` trotzdem aktualisiert wird.

4.2.14 Prozess der Bildauswahl im Chat

Listing 9: Prozess der Bildauswahl und -verarbeitung

```

1
2  void _handleImageSelection(BuildContext context) async {
3    var result = await ImagePicker().pickImage(
4      imageQuality: 70,
5      maxWidth: 1440,
6      source: ImageSource.gallery,
7    );
8
9    final image = await result!.readAsBytes();
10   if (image == null) {
11     Get.snackbar("null", "null");
12   }
13
14  if (result != null) {
15    _setAttachmentUploading(true);
16    final file = File(result.path);
17    final size = file.lengthSync();
18    final bytes = await result.readAsBytes();
19    final image = await decodeImageFromList(bytes);
20    final name = result.name;
21
22    try {
23      String userId = room.users[0].id;
24      final reference = FirebaseStorage.instance.ref('$userId/$name');
25      await reference.putFile(file);
26      final uri = await reference.getDownloadURL();
27
28      final message = types.PartialImage(
29        height: image.height.toDouble(),
30        name: name,
31        size: size,
32        uri: uri,
33        width: image.width.toDouble(),
34      );
35
36      chat.FirebaseAuthCore.instance.sendMessage(
37        message,
38        room.id,
39      );
40      _setAttachmentUploading(false);
41    } finally {
42      _setAttachmentUploading(false);
43    }
44  }
45 }
```

Die Funktion `handleImageSelection` wählt ein Bild aus der Galerie des Benutzers aus,

lädt es in den Firebase-Speicher hoch und sendet es als Nachricht in den Chatraum.

Die Schritte, die die Funktion ausführt, werden wie folgt ausgeführt:

Die Funktion verwendet `ImagePicker().pickImage`, um die Galerie des Benutzers asynchron zu öffnen und dem Benutzer die Möglichkeit zu geben, ein Bild mit der angegebenen `imageQuality` und `maxWidth` auszuwählen. Das ausgewählte Bild wird in der Ergebnisvariablen gespeichert.

Es liest das Bild als Bytes und prüft, ob es null ist. Wenn dies der Fall ist, wird eine Snackbar mit *null* als Warnung angezeigt.

Wenn das Ergebnis nicht null ist, dann macht die Funktion mit den folgenden Schritten weiter:

Der Status des Uploads von Anhängen wird mit der Funktion *setAttachmentUploading* auf *true* gesetzt.

Ein File wird aus dem Pfad des Bildes erstellt, und die Größe wird mit *lengthSync()* berechnet.

Die Bytes des Bildes werden gelesen, und das Bild wird aus der Byteliste dekodiert.

Der Name des Bildes wird aus dem *result* object ausgelesen.

In einem try-final-Block:

Die ID des Benutzers wird aus der Liste *room.users* abgerufen.

Ein Referenz zu dem Ort in Firebase Storage, an dem das Bild gespeichert wird, wird erstellt, indem die ID des Benutzers und der Name des Bildes verwendet werden.

Die Datei wird mit *putFile* in Firebase Storage hochgeladen. Die Download-URL der hochgeladenen Datei wird von Firebase Storage abgerufen. Es wird eine *types.PartialImage*-Nachricht mit den Merkmalen des Bildes (Höhe, Breite, Name, Größe und URI) erstellt. Die Bildnachricht wird mit *chat.FirebaseChatCore.instance.sendMessage* an den Chatraum gesendet. Der Upload-Status des Bildes wird auf *false* gesetzt.

Im final-Block stellt die Funktion sicher, dass der Hochladestatus des Anhangs auf *false* gesetzt wird, auch wenn ein Fehler während des Vorgangs auftritt. Dadurch wird sichergestellt, dass die Benutzeroberfläche im Falle von Fehlern unverändert bleibt.

4.2.15 Indexing-Problem

Listing 10: Anzeige des Profilbildes

```

1  Image displayProfileImage() {
2      String fullname = "${room.users[0].firstName} ${room.users[0].lastName}";
3      if (room.name == fullname) {
4          return Image.network('${room.users[0].imageUrl}');
5      } else {
6          return Image.network('${room.users[1].imageUrl}');
7      }
8  }

```

Das *Indexing-Problem* entsteht, wenn man versucht, auf ein Element mit einem Index zuzugreifen, der in einem Array oder einer Liste nicht vorhanden ist. Um solche Fälle zu vermeiden, müssen Entwickler sicherstellen, dass gültige Indexe geprüft werden, bevor auf Elemente in einer Liste oder einem Array zugegriffen wird.

Im angegebenen Beispiel behandelt die Methode *displayProfileImage()* ein spezielles Indexierungsproblem:

Aufbau eines *fullname*-Strings durch Verkettung des *firstName* und *lastName* des Benutzers bei Index 0 in der Liste *room.users*. Zuerst wird geprüft, ob *room.name* gleich dem *fullname* ist: Wenn ja, wird das Profilbild des Benutzers bei Index 0 angezeigt, indem ein *Image.network* Widget mit der *imageUrl* des Benutzers erstellt wird. Wenn false, wird davon ausgegangen, dass das gewünschte Profilbild das des Benutzers bei Index 1 ist und es wird ein *Image.network* Widget mit der *imageUrl* des Benutzers bei Index 1 geliefert.

Google Smart Reply ML Kit

Die Implementierung des Google Smart Reply ML Kits wurde als sinnvolle Ergänzung in Betracht gezogen, um wiederkehrende und standardisierte Antworten in der Kommunikation zwischen Benutzern zu reduzieren. Ein Beispiel dafür ist die Nutzung solcher Plattformen wie Willhaben, bei denen häufig ähnliche Fragen und Antworten ausgetauscht werden.

Die Integration des Google Smart Reply ML Kits bietet den Benutzern verschiedene Antwortmöglichkeiten, die direkt über den Eingabefeld des Chats angezeigt werden. Dies ermöglicht eine effizientere und benutzerfreundlichere Kommunikation, da aufwendige manuelle Texteingaben vermieden werden können. Die Abbildung verdeutlicht, wie verschiedene Vorschläge im Eingabefeld angezeigt werden.

Das Google Smart Reply System nutzt die letzten zehn Nachrichten als Kontext, um adäquate und relevante Antwortvorschläge zu generieren. Die Anwendung von maschinellem Lernen in diesem Kontext trägt zur Verbesserung der Benutzererfahrung bei und ermöglicht eine effizientere Kommunikation. Auf diese Weise können die Benutzer schnell und einfach auf eingehende Nachrichten reagieren, wodurch wertvolle Zeit gespart und mögliche Frustrationen vermieden werden.

4.2.16 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
- Seit wann in der Nachbarschaft
- Beruf
- Bio
- Interessen
- Bietet an

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

Die Abbildung 28 zeigt, wie das öffentliche Profil in der App dargestellt wird.

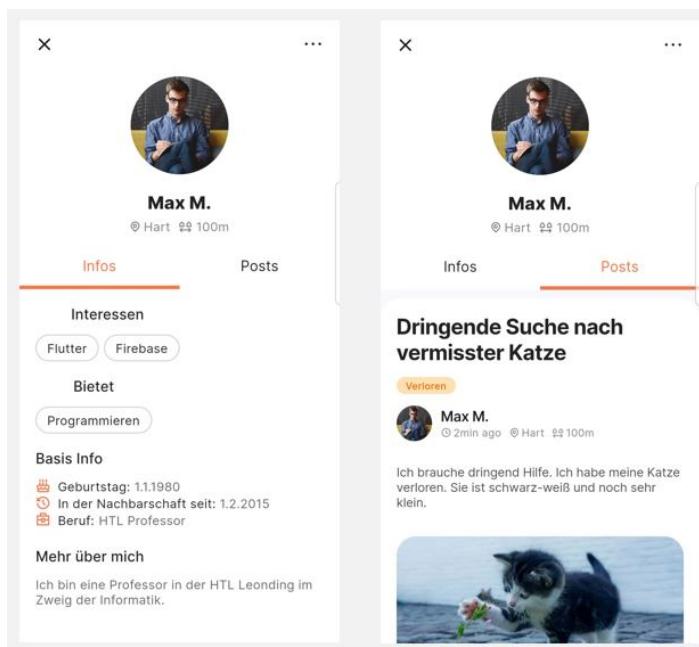


Abbildung 28: Öffentliche Profilansicht auf der App

Profil bearbeiten

Das öffentliche Profil eines Users kann von diesem bearbeitet werden.

Die Abbildung 29 zeigt, wie die Profilbearbeitung in der App dargestellt wird.

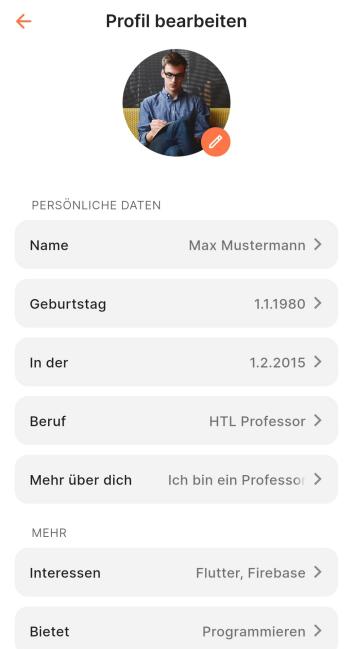


Abbildung 29: Profilbearbeitungsansicht auf der App

Profil Melden

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

Die Abbildung 30 zeigt, wie das Melden eines Profils in der App dargestellt wird.

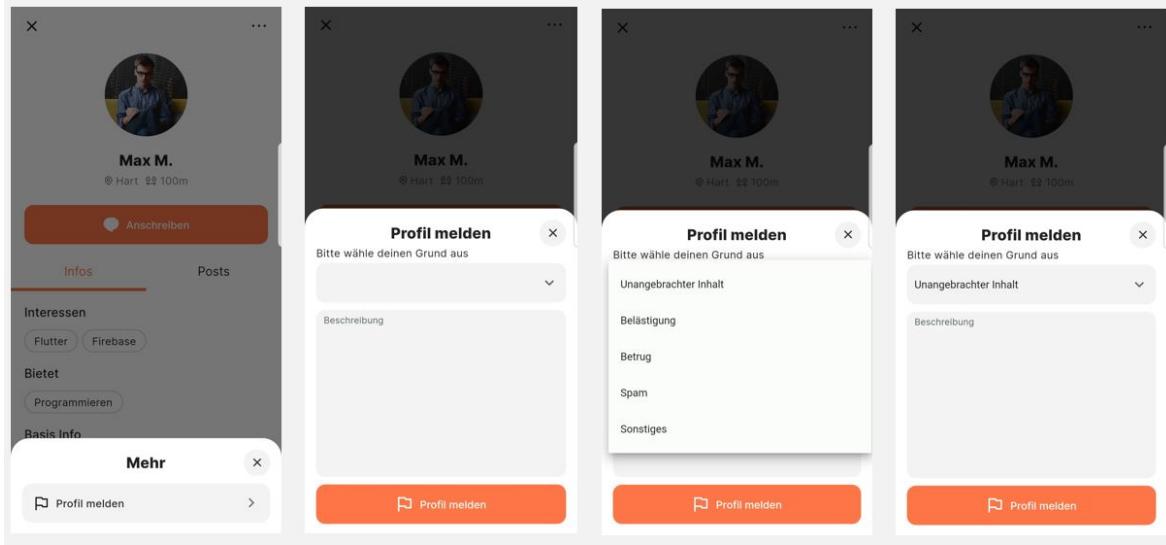


Abbildung 30: Meldeansicht für ein Profil auf der App

4.2.17 Benachrichtigungen

Benachrichtigungen dienen dazu die Nachbarn über mögliche Hilfsangebote 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 einer „Annehmen“- und „Ablehnen“-Option ausgestattet, womit das Hilfsangebot angenommen oder abgelehnt werden kann. Die Abbildung 31 zeigt, wie die Benachrichtigungen in der App dargestellt werden.

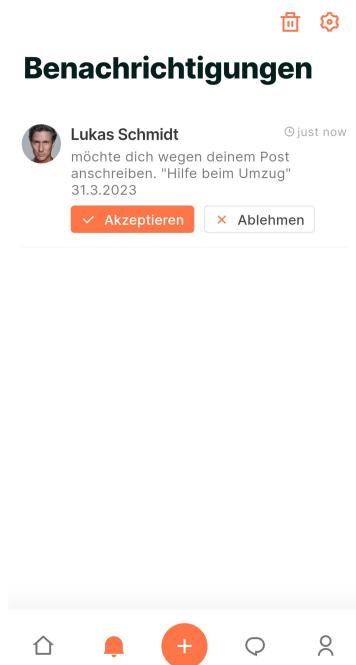


Abbildung 31: Benachrichtigungsansicht auf der App

4.2.18 Einladecode

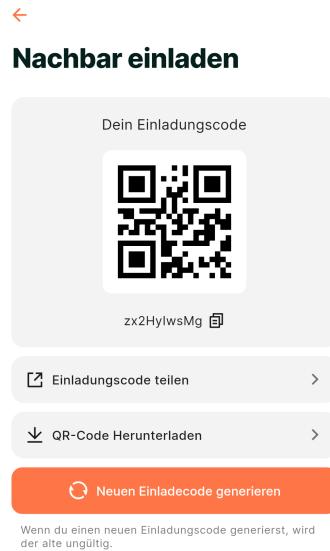


Abbildung 32: Einladecode-Seite in der App

Eine der beiden Verifizierungsmethoden basiert auf der Nutzung von Einladungscodes. Als Inspiration diente die deutsche App *Nebenan*, die ebenfalls eine Verifizierung über einen Code ermöglicht. Da das Team dies als gute Lösung erachtete, wurde ein ähnlicher Ansatz übernommen.

Jeder Code hat eine Reichweite von 1 km, ausgehend von der Adresse des Benutzers, der den Code generiert. Dies soll verhindern, dass ein Code missbräuchlich verwendet wird. Der Aktionsradius, in dem der Benutzer den Code nutzen kann, kann von den Administratoren variabel eingestellt werden.

Zudem wird für die Entwickler protokolliert, wer welchen Code zur Verifizierung verwendet hat, um im Falle eines Missbrauchs entsprechende Maßnahmen ergreifen zu können, wie beispielsweise das Sperren des Codes. Administratoren haben ebenfalls die Möglichkeit, Codes zu deaktivieren.

Einladungsmöglichkeiten

Folgende Punkte sind in der Abbildung 32 zu erkennen.

1. Generierung eines QR-Codes:

- Beim Öffnen der Seite *Nachbar einladen* wird automatisch ein QR-Code generiert.

- Nutzer können den generierten Code in die Zwischenablage kopieren, indem sie den Button neben dem Code betätigen.

2. Teilen des Einladungscodes:

- Bei dem Button *Einladungscode teilen* wird eine Nachricht mit dem generierten Code erstellt.
- Der Nutzer kann die Nachricht mithilfe des systemspezifischen Sharesheets teilen.

3. Herunterladen des QR-Codes:

- Unter *QR-Code herunterladen* wird eine PNG-Datei des Einladungs-QR-Codes erstellt.
- Die Datei wird im systemspezifischen Sharesheet geöffnet, um sie speichern zu können.

4. Generierung eines neuen Einladungscodes:

- Der orangefarbene Button *Neuen Einladungscode generieren* ermöglicht das Erstellen eines neuen Codes.
- Sobald der Nutzer einen neuen Code erstellt, wird der vorherige Code deaktiviert.

Verifizierte Nutzer können alle 24 Stunden einen Einladungscode generieren.

Die Verifizierung erfolgt, indem der Benutzer bei der Registrierung die Option *Mit Einladungscode verifizieren* auswählt. Anschließend hat er die Möglichkeit, den Einladungscode entweder über den integrierten QR-Code-Scanner zu scannen oder manuell einzugeben.

Für die Implementierung des QR-Code-Scanners wurde das Flutter-Paket [45] mobile_scanner verwendet, welches auf Googles MLKit [48] basiert.

MLKit ist eine Machine-Learning-Bibliothek, die im vorliegenden Fall für das zuverlässige und schnelle Scannen von QR-Codes eingesetzt wird. Diese Bibliothek ermöglicht es, komplexe Funktionen wie das Erkennen von QR-Codes effizient und präzise zu implementieren. Der einzige Nachteil dieses Pakets ist der hohe Speicherbedarf von etwa 10 MB. Daher ist in Zukunft geplant, das MLKit nur bei Bedarf herunterzuladen und nach erfolgreicher Verifizierung des Benutzers wieder zu entfernen.

Der gescannte oder eingegebene Code wird anschließend an die Cloud-Funktion `checkVerification` (siehe Abschnitt 4.5.5) gesendet und überprüft, ob der Code korrekt ist.

4.2.19 Einstellungen

Die Einstellungssicht der Anwendung bietet eine Reihe an nützlichen Funktionen, die dem User mehr Kontrolle über sein Konto geben.

Dazu gehört unter anderem 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 Option 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.

4.2.20 Feedback

Während der Testphase besteht das Ziel darin, eine effiziente Methode zur Sammlung von Feedback zu finden. Für ein detailliertes Verständnis von Fehlermeldungen ist die Bereitstellung von Screenshots äußerst hilfreich. Aus diesem Grund wurden Packages untersucht, die das Erstellen von Screenshots ermöglichen und zusätzliche Informationen als Kontext hinzufügen können. Das ausgewählte Package ist [49]. Dieses erlaubt es den Nutzern, Screenshots anzufertigen, Annotationen hinzuzufügen und anschließend einen entsprechenden Textkontext einzufügen. Eine nützliche Funktion ermöglicht es den Nutzern, bei Bedarf erneut zu navigieren und den Screenshot zu wiederholen.

Um den Feedback-View zu initiieren, wurde das Package [50] shake verwendet. Dieses wird durch eine bestimmte Bewegung des Mobilgeräts aktiviert. Bei der Implementierung dieser Funktion wurden ähnliche Mechanismen berücksichtigt, die in anderen Anwendungen zur Feedback-Erhebung verwendet werden.

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[51], Instagram[52] und TikTok[53] 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[54], 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 Dribbble[55] und Mobbin[56] 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.

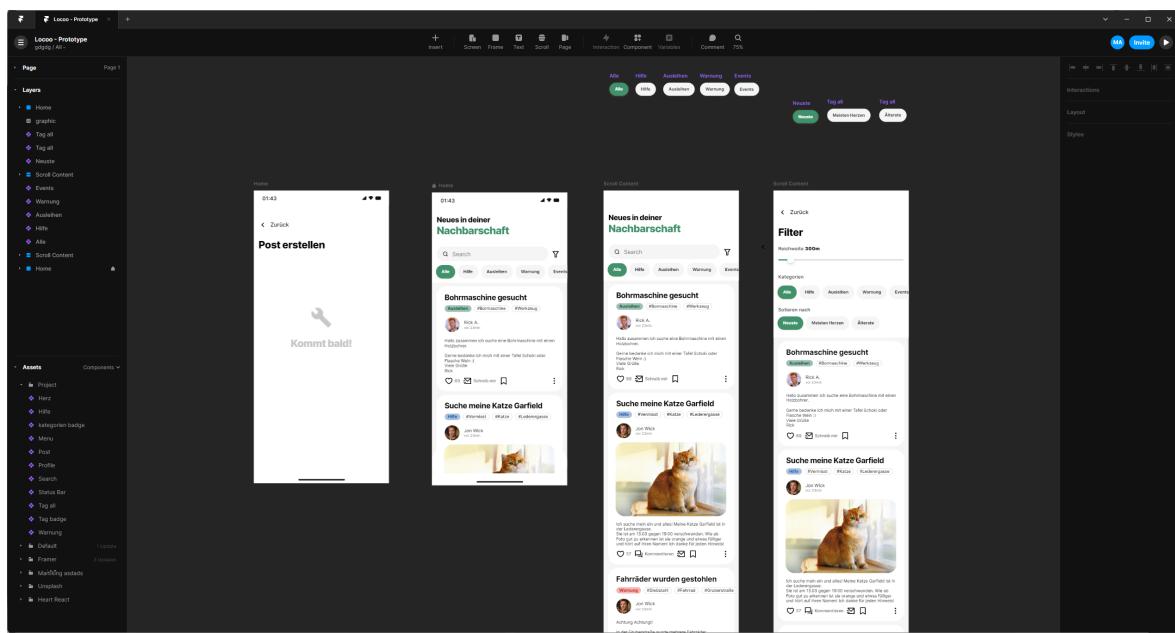


Abbildung 33: Nochba Prototyp in Framer

Framer wurde während des Hackerthon Linz haCKt verwendet, um den ersten Prototypen zu erstellen wie bei der man bei der Abbildung 33 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 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 34 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.

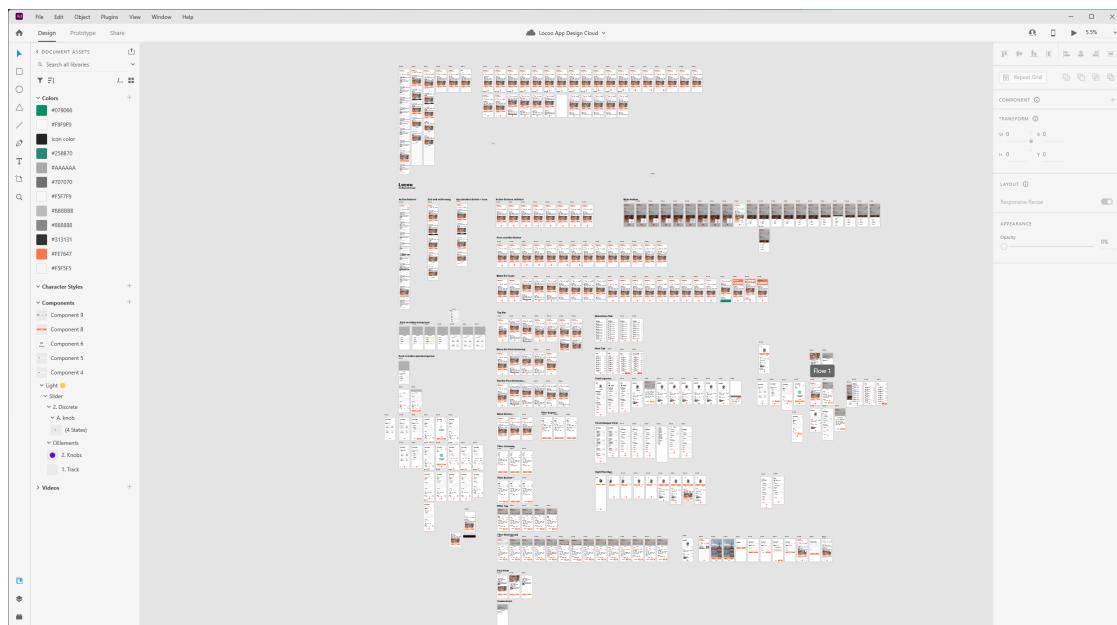


Abbildung 34: Nochba Prototyp in Adobe XD

Die Abbildung 34 zeigt eine Sammlung von Designs und Designentwicklungen im Prototypen. Adobe XD erwies sich als passend für das Projekt, da Martin Hausleitner, der alleinige Designer, keine Kollaborationsfunktionen benötigte. Das Tool ermöglichte es ihm, verschiedene Designvarianten effizient zu entwickeln und zu optimieren, um eine ansprechende Benutzeroberfläche für die Anwendung zu erstellen.

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

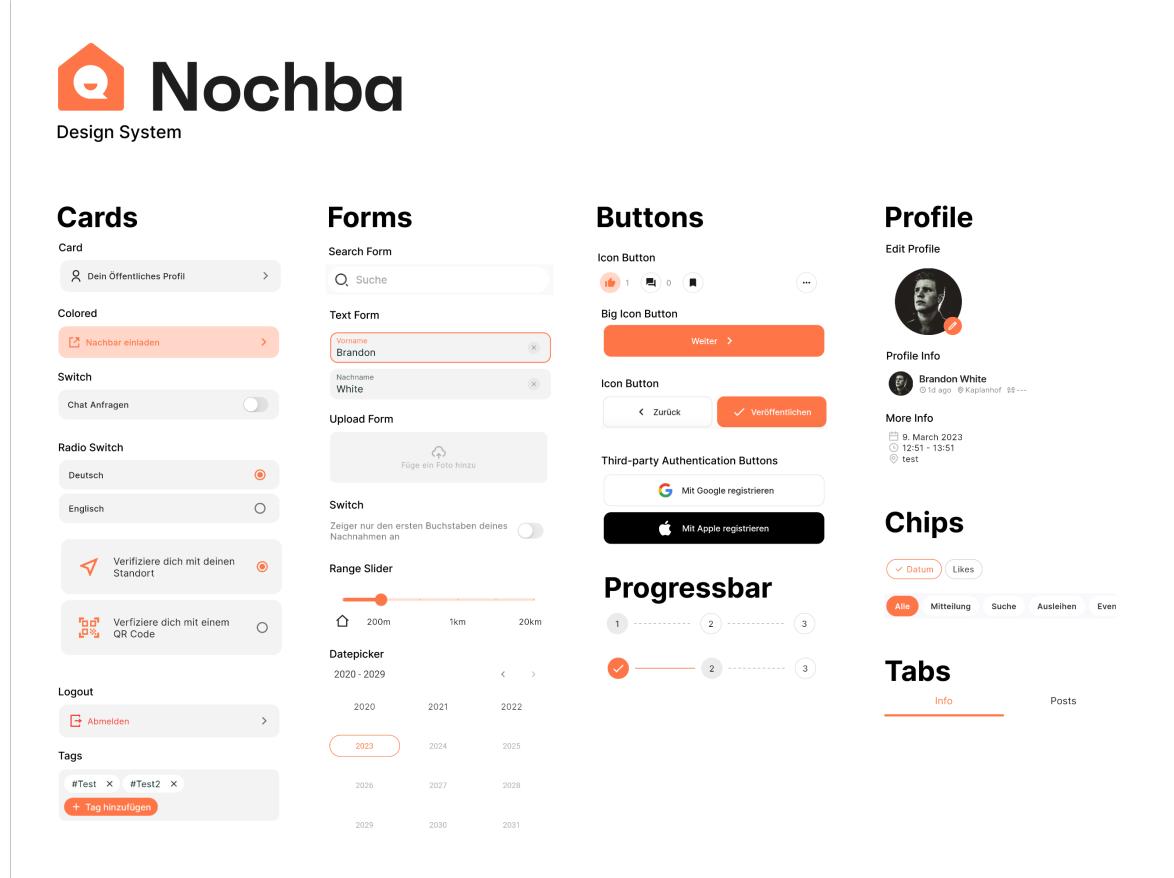


Abbildung 35: Nochba Design System

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 35 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.

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 Dateipicker wurde die Bibliothek Syncfusion flutter datepicker [57] 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.

In der App wurden zusätzlich Third-Party-Anmeldemöglichkeiten Buttons integriert. Dazu gehören das Login mit Apple und das Login mit Google. Um eine konsistente Benutzeroberfläche sicherzustellen, wurden diese Anmeldeoptionen entsprechend an das Design angepasst.

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 kommen in der App häufig zum Einsatz, um Filter auszuwählen. Der Designer ist allerdings mit dem finalen Design noch nicht vollständig zufrieden, da es zahlreiche unterschiedliche Varianten gibt. Die Gestaltung der Chips wird weiterhin optimiert, um eine einheitliche und ästhetisch ansprechende Darstellung zu erreichen. Dieser fortlaufende Prozess zeigt das Bestreben, die Benutzeroberfläche kontinuierlich zu verbessern und den Anforderungen der Benutzer gerecht zu werden.

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

S 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.

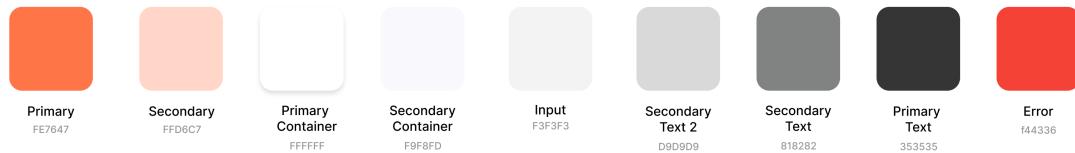


Abbildung 36: Nochba Farbpalette

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 36 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 37: App Icons

Die obige Abbildung 37 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[58] 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[59] gibt, was die Implementierung erleichtert hat.

Obwohl Remixicons viele Icons zur Verfügung stellt, wurden bei einigen Icons die Material Icons[60] 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.

Inter fügt über eine hohe x-Höhe, um die Lesbarkeit von Text in Groß- und Kleinbuchstaben zu verbessern. Mehrere OpenType-Funktionen stehen ebenfalls zur Verfügung, z. B. kontextabhängige Alternativen, die die Zeichensetzung an die Form der umgebenden Glyphen anpassen, geschlitzte Nullen, wenn Sie "0" von "o" unterscheiden müssen, tabellarische Zahlen, usw.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
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 Regular Sample

Dynamic Metrics

Inter Bold Sample

ABCDEFGHIJKLMN
HIJKLMNOP
QRSTUVWXYZ
YZ1234567
890aåbcde
fghijklmnop
qrstuvwxyz.

The method by means of which the musical signals may be sent simultaneously in both directions is shown in figures 8, 9, and 10. The arrangement is similar to that shown in figures 4, 5, and 6, excepting that the intermission current from 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 secured at either end of the instrument. The receiving instrument receives manipulation of any key are received at the common terminal, and the number of signals produced by the manipulation of any key are received at every station. It was also found to practical upon either of the plans here shown, to transmit signals from one instrument to another, without the necessity of connecting the two instruments directly at every station. It was also found to practical upon either of the plans here shown, to transmit signals from one instrument to another, without the necessity of connecting the two instruments directly at every station.

The method by means of which the musical signals may be sent simultaneously in both directions is shown in figures 8, 9, and 10. The arrangement is similar to that shown in figures 4, 5, and 6, excepting that the intermission current from 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 secured at either end of the instrument. The receiving instrument receives manipulation of any key are received at the common terminal, and the number of signals produced by the manipulation of any key are received at every station. It was also found to practical upon either of the plans here shown, to transmit signals from one instrument to another, without the necessity of connecting the two instruments directly at every station. It was also found to practical upon either of the plans here shown, to transmit signals from one instrument to another, without the necessity of connecting the two instruments directly at every station.

Abbildung 38: Die Schriftfamilie Inter

Ein wichtiger Teil jedes Designs ist die Typografie, so auch bei der Nachba-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 [61], die man auf der Abbildung 38 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 [61]. 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

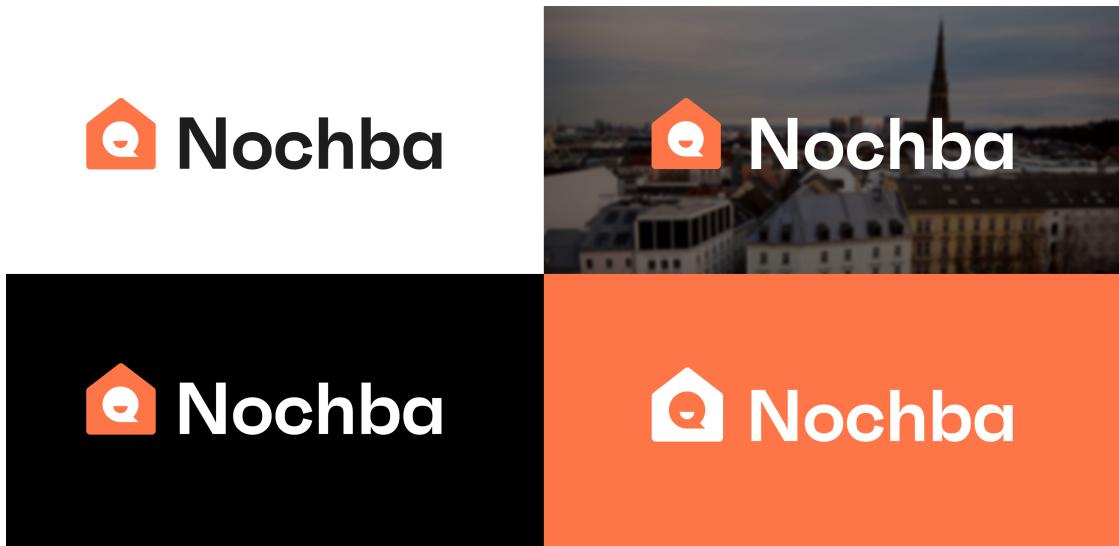


Abbildung 39: Finales Logo

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.



Abbildung 40: Logo Design Ideen

Die endgültige Gestaltung des Logos hat viel Zeit in Anspruch genommen, da die vorherigen Entwürfe nicht vollständig zufriedenstellten die man bei Abbildung 40 sieht. Nach anderen Logos, die eine Verbindung zu Nachbarschaften oder Häusern herstellen, wurde gezielt gesucht 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 39 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, verspielte Schriftart wurde bewusst gewählt, da sie besser mit der Farbmischung harmoniert als die vorherige Schriftart.

4.3.3 App Design

Thumb Zone Prinzip

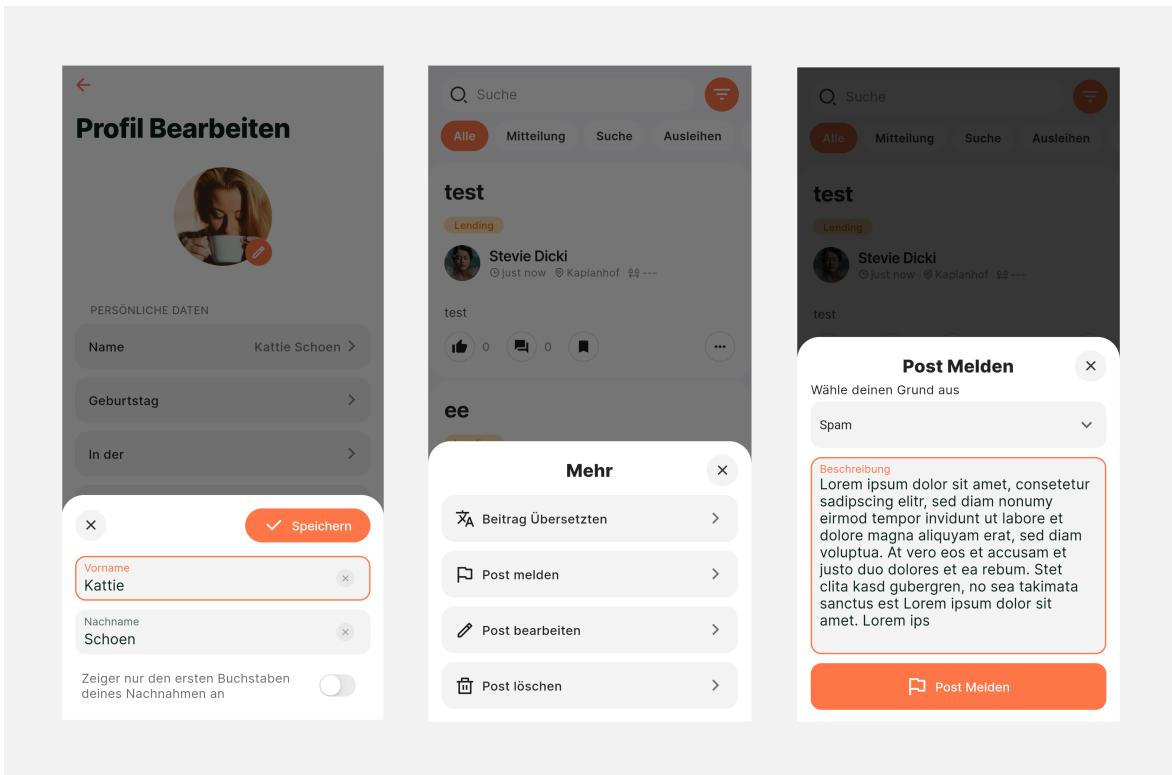


Abbildung 41: Abbildung mehrere Thumbs Zones in der App

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 41 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 41 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 zukünftigen Versionen der App soll sichergestellt werden, dass diese Prinzipien einheitlich angewendet werden.

Design Historie

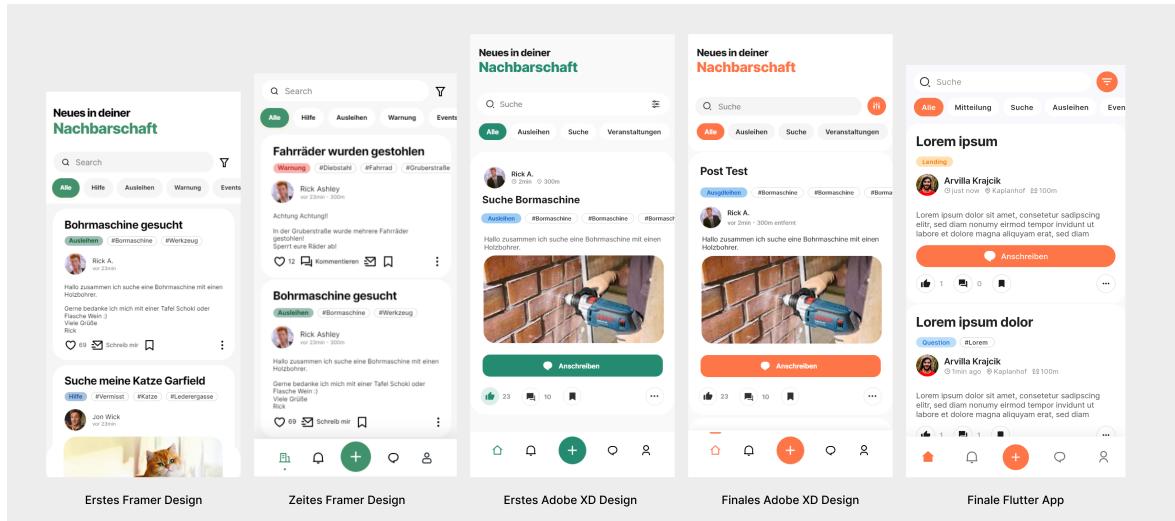


Abbildung 42: Screenshots verschiedener Versionen des App-Home-Screen

Im Rahmen des Hackathons Linz hACkT wurde das erste Design in Abbildung 42 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 42 und dem ersten Adobe XD-Design 42 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 42 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

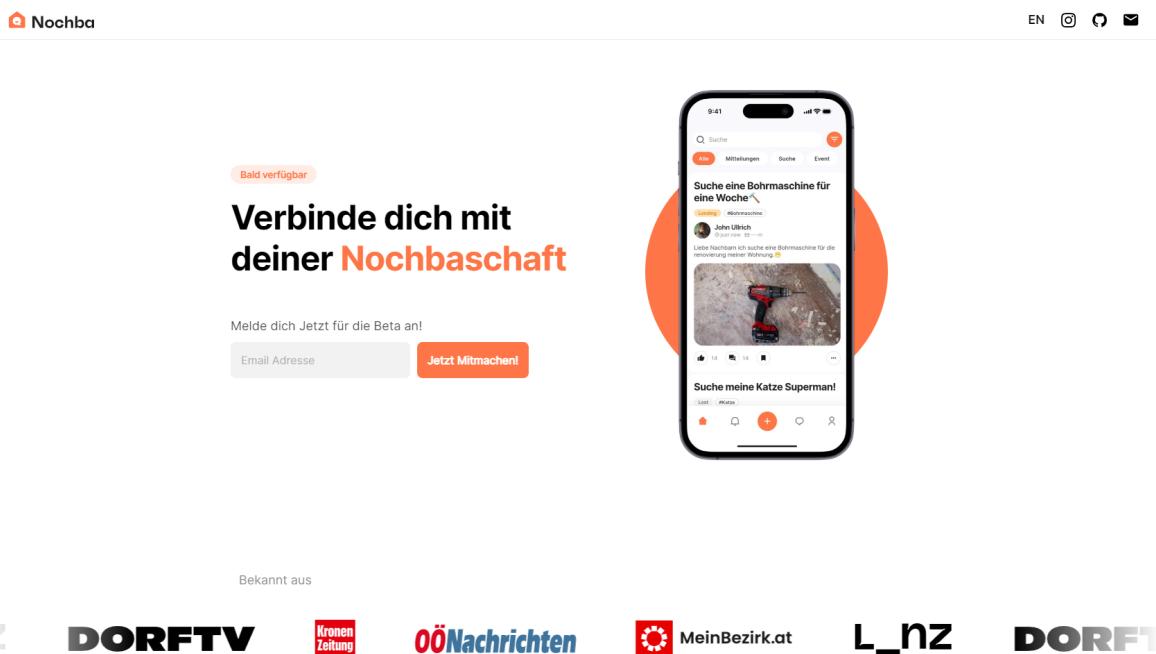


Abbildung 43: Screenshot der Nochba Website: nochba.at

Die Medienpräsenz des Projekts, die durch die Teilnahme an Linz hACkT erlangt wurde, und das Fehlen eines zentralen Anlaufpunkts für Informationen führten zur Entscheidung, eine einfache Landing Page zu erstellen. Diese Seite bietet grundlegende Informationen und ist unter nochba.at[62] erreichbar.

Die Teilnahme am mPreneur Social Mobile Entrepreneurship von Arsham Edalatkahah führte zu internationaler Aufmerksamkeit für das Projekt. Dies veranlasste die Erstellung einer englischsprachigen Website unter nochba.com[63].

Ziel ist es, die Arbeit einem breiteren Publikum zugänglich zu machen und die Reichweite des Projekts zu erhöhen. Durch die Präsenz in einer internationalen Sprache kann das Interesse von Menschen aus verschiedenen Ländern und Kulturen geweckt und potenziell eine größere Nutzerbasis erreicht werden.

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[64]
 - Linz[65]
 - OÖNachrichten[66]
 - DorfTV[67]
 - 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[68]

- Instagram[69]
- Email project@nochba.com

4.4.4 Hosting

Die Website wurde mithilfe von [70] erstellt und auf deren Hosting-Plattform gehostet. Um die Domain-Namen zu nutzen, die über [71] (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

Aktuell verfügt die Website über kein SSL-Zertifikat, da durch die Maskierung das Framer SSL-Zertifikat entfällt. Dennoch wird die Website im Browser nicht als bedrohlich eingestuft. Aus diesem Grund wurde entschieden, kein SSL-Zertifikat zu erstellen und somit den Aufwand zu reduzieren.

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 Authentication

Firebase Authentication ist ein wichtiger Bestandteil für die Integration von Benutzerauthentifizierung in der Applikation. Die Implementierung dieses Dienstes trägt zur Einhaltung der Privatsphäre der User bei, indem ein sicherer und zuverlässiger Authentifizierungsmechanismus bereitgestellt wird.

Darüber hinaus ermöglicht der Dienst eine unkomplizierte Integration mit anderen Firebase-Services, wie zum Beispiel Cloud Firestore.

Für die vorliegende Applikation wurden zwei Authentifizierungsmethoden ausgewählt, die über Firebase Authentication bereitgestellt werden: E-Mail und Passwort sowie Google Authentication. Die Wahl dieser beiden Methoden basiert auf ihrer weiten Verbreitung und Akzeptanz unter den Usern. Beide Methoden bieten unterschiedliche Vorteile und ermöglichen den Benutzern, je nach ihren Präferenzen und Anforderungen, eine geeignete Anmeldeoption auszuwählen.

4.5.3 Cloud Firestore

Cloud Firestore dient der Anwendung als NoSQL-Datenbank, die die dokumentbasierte Speicherung von Daten ermöglicht. Dadurch kann das Datenmodell an die Anforderungen der Anwendung zugeschnitten werden, während es trotzdem effizient und performant bleibt.

Ein wichtiger Vorteil von Firestore ist, dass Collections und dazugehörige Subcollections erstellt werden können, um damit Daten in hierarchischen Strukturen zu organisieren. Diese Subcollections können verwendet werden, um abhängige Daten in einer einzigen Anfrage abzurufen und damit den Datenzugriff zu optimieren. Darüber hinaus bietet Firestore die Möglichkeit, Daten auf verschiedenen Ebenen zu filtern und zu sortieren, um nur die benötigten Daten abzurufen, was beim Filtern von Beiträgen eine große Rolle spielen kann.

Ein weiterer Vorteil ist die Skalierung von Daten auf Firestore, womit die Anwendung mit hohen Nutzerzahlen und großen Datenmengen, wie zum Beispiel dem Abrufen von Beiträgen, zurecht kommen kann.

Außerdem bietet Firestore noch die Möglichkeit an, Daten in Echtzeit zu synchronisieren, womit die Anwendung sofort auf Änderungen in der Datenbank reagieren und somit reaktiv bleiben kann.

Datenmodell

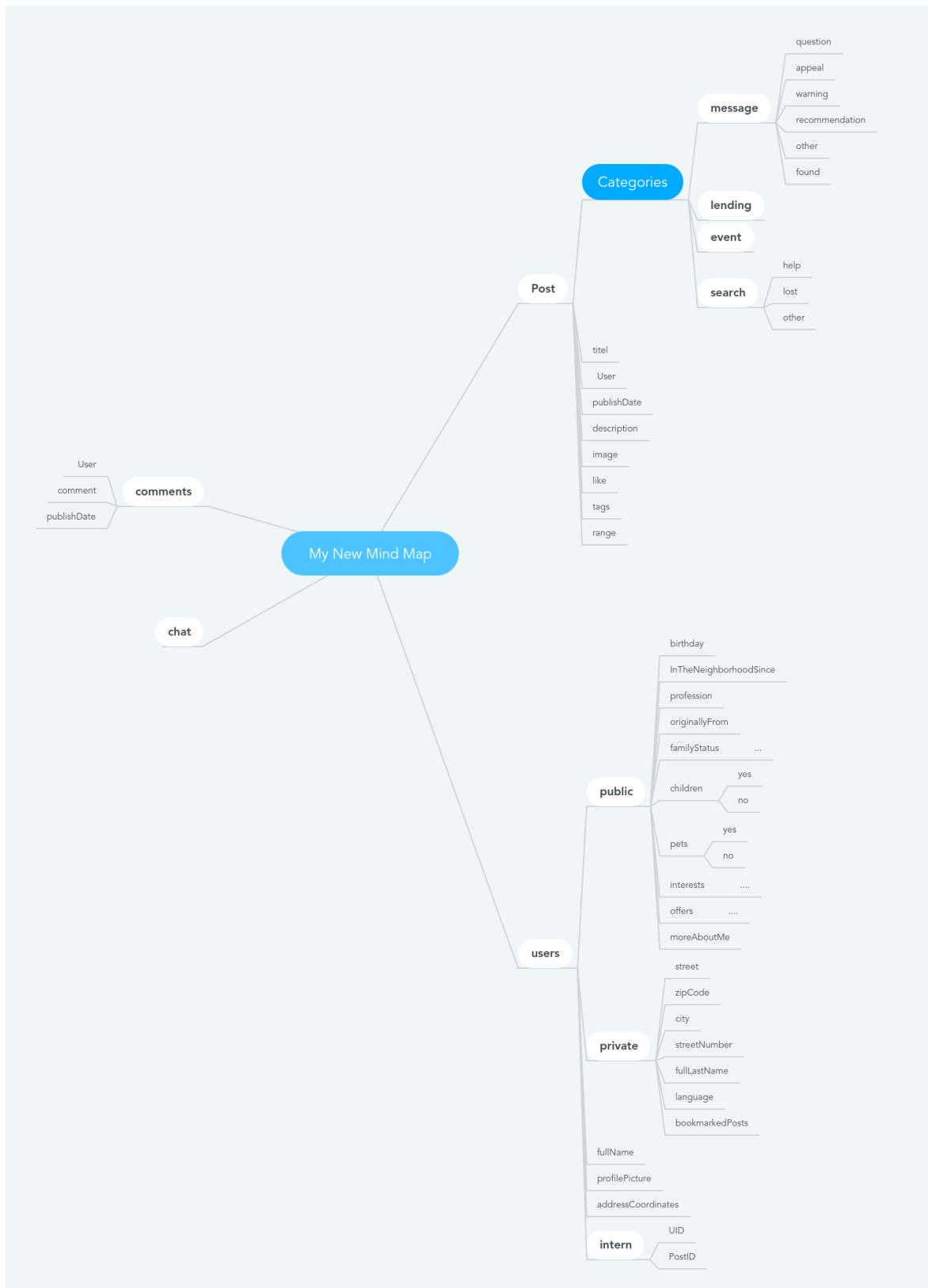


Abbildung 44: Das erste Datenmodell

Abbildung 44 repräsentiert den Ansatz des ersten Datenmodells, welches im Zuge der Diplomarbeit erstellt wurde. Da das Team zum Zeitpunkt des Erstellens des Daten-

modells über keine praktische Erfahrung mit Firestore oder mit NoSQL-Datenbanken verfügte, ist das Datenmodell dementsprechend unvollständig und lückenhaft.

Es besteht aus vier Hauptsammlungen: **users**, **posts**, **comments** und **chats**

Users

Die **users**-Collection ist die Sammlung, die die Daten des Users aufbewahren sollte. Die Dokumente der Collection beinhalten die Felder **fullName** für den Anzeigename, **profilePicture** für das Profilbild und **addressCoordinates** für die Koordinaten des Users. Der Anzeigename ist der Name der den anderen Usern angezeigt wird und kann sich entweder aus dem Vor- und Nachnamen zusammensetzen (Max Mustermann) oder aus dem Vornamen und Initial des Nachnamen (Max M.). Weiters beinhaltet jedes Dokument die drei weiteren Sub-Collections **public**, **private** und **intern**.

Die **public**-Collection beinhaltet öffentliche Informationen von dem User, die für jeden anderen User zugänglich sind. Die Dokumente der Collection beinhalten die folgenden Felder:

- **birthday** für den Geburtstag des Nachbarn oder der Nachbarin
- **InTheNeighbourhoodSince** für das Datum seitdem der Nachbar oder die Nachbarin in der Nachbarschaft ist
- **Profession** für den Beruf des Nachbarn oder der Nachbarin
- **originallyFrom** für den Heimatsort des Nachbarn oder der Nachbarin
- **familyStatus** für den Familienstatus des Nachbarn oder der Nachbarin
- **children** ob der Nachbar oder die Nachbarin Kinder hat
- **pets** ob der Nachbar oder die Nachbarin Haustiere hat
- **interests** für die Interessen des Nachbarn oder der Nachbarin
- **offers** für was angeboten werden kann von dem Nachbarn oder der Nachbarin
- **moreAboutMe** für eine genauere Selbst-Beschreibung des Nachbarn oder der Nachbarin

Diese Daten werden vom User selbst festgelegt und dann beim öffentlichen Profil angezeigt.

Die `private`-Collection beinhaltet private Informationen von dem User, zu denen kein anderer Zugriff hat. Daten die in diese Sammlung gespeichert werden, sind die Adresse mit Hilfe der Felder `street`, `streetNumber`, `city` und `zipCode`, der vollständige Nachname des Users mit dem Feld `fullLastName`, die eingestellte Sprache mit dem Feld `language` und die vom User durch Markierung gespeicherten Beiträge mit dem Feld `bookMarkedPosts`.

Die `intern`-Collection beinhaltet interne Informationen von dem User, die für die saubere Funktionalität der App wichtig sind, wie die Benutzeridentifikationsnummer mit dem Feld `uid` und die geliketen Posts des Users mit dem Feld `postId`

Posts

Die `posts`-Collection ist die Sammlung, die die Beiträge der User aufbewahren soll. Die Dokumente der Collection beinhalten die Felder `titel` für den Titel, `description` für die Beschreibung, `image` für das anhängbare Bild, `tags` für die anhängbaren Tags, `range` für die Reichweite innerhalb der Beitrag nur gesehen werden soll, `like` für den Like-Zähler, `user` für die Benutzer-Id von dem der Beitrag stammt und `publishDate` für das Veröffentlichungsdatum. Als weiteres Feld zählt noch `category`, welches im DatenModell als eine Enumeration dargestellt wird.

Comments

Die `comments`-Collection ist die Sammlung, die die Kommentare der Beiträge aufbewahren soll. Die Dokumente der Collection beinhalten die Felder `comment` für den Text, `user` für die Benutzer-Id von dem der Kommentar stammt und `publishDate` für das Veröffentlichungsdatum. In diesem Ansatz wurde auf die Id des Beitrags vergessen, unter welchem der Kommentator hinterlassen wurde.

Chats

Obwohl die `chats`-Collection im ursprünglichen Datenmodell enthalten ist, sind die zugehörigen Felder und Strukturen nicht weiter definiert.

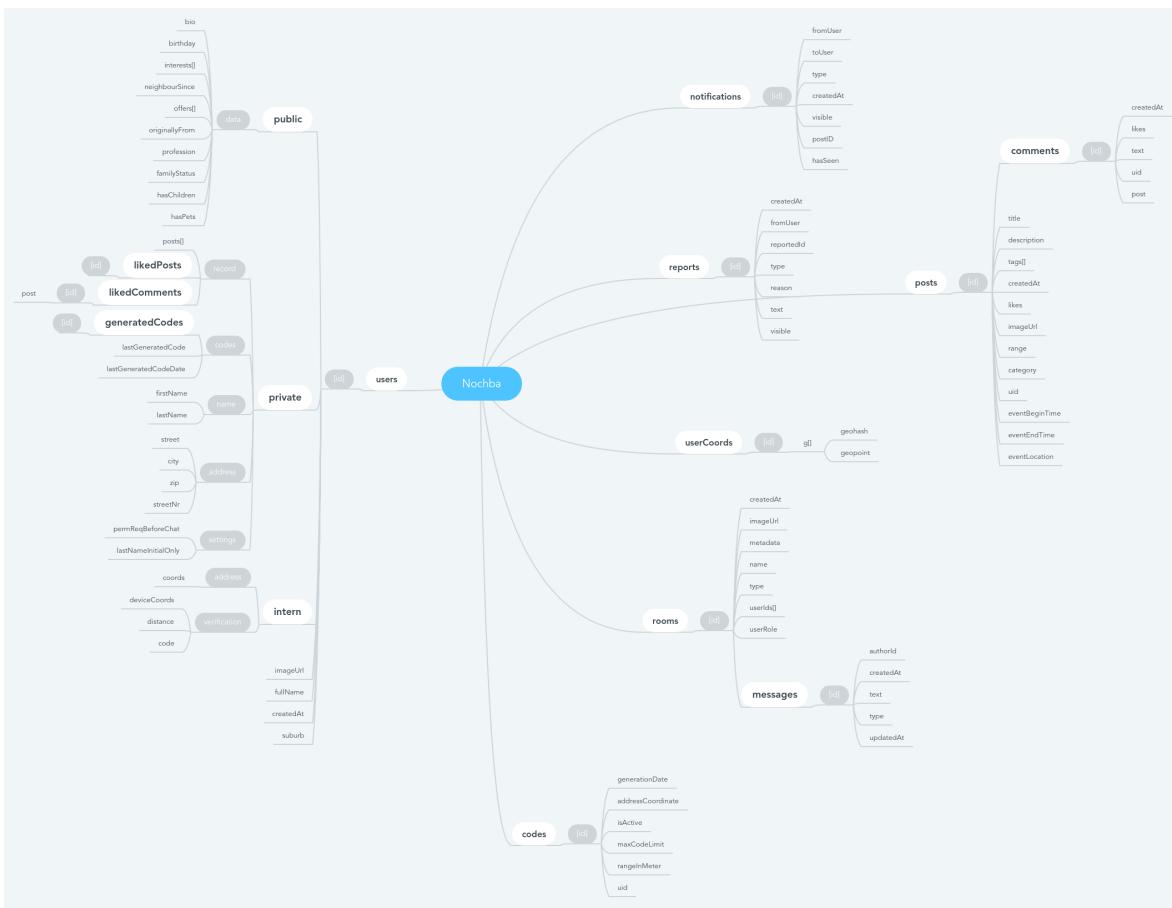


Abbildung 45: Das neue Datenmodell

Abbildung 45 repräsentiert das aktuelle Datenmodell. Durch das erweiterte Know-How, welches sich das Team im Lauf der Diplomarbeit angeeignet hat, wurde darauf geachtet eine möglichst effiziente Datenaufteilung und Datenspeicherung zu gestalten.

Um das zu gewährleisten wurden folgende Prinzipien erstellt:

1. Falls eine Collection abhängig von einer anderen Collection ist, wird diese der anderen Collection als Sub-Collection hinzugefügt - Beispiel: Beiträge und Kommentare
2. Daten, die miteinander nichts zu tun haben werden in eigene Dokumente gespeichert
3. Falls eine Collection einmalig verschiedene Daten speichern soll, so bekommen die Dokumente eine vorbestimmte Id, um das Verwalten der Dokumente zu erleichtern - Beispiel: Die **private**-Subcollection innerhalb der **users**-Collection beinhaltet das Dokument **name**, wo der Name gespeichert wird und das Dokument **address**, wo die Addressdaten gespeichert werden

Im Gegensatz zum alten Datenmodell wurde das neue Datenmodell um mehrere Collections erweitert und manche Collections wurden angepasst. Es besteht aus folgenden Hauptsammlungen: `users`, `posts`, `notifications`, `reports`, `codes` und `chats`

Users

Die `users`-Collection ist die Sammlung, die die Daten des Users aufbewahrt. Die Dokumente der Collection beinhalten wie beim alten Datenmodell die Felder `fullName` für den Anzeigename und `profilePicture` für das Profilbild. Das Feld `addressCoordinates` für die Koordinaten des Users wurde entfernt, weil es sich als Datenleck von privaten Userdaten herausstellte. Die Felder der `users`-Collections dienen als Anzeigedaten, von daher wurde das `addressCoordinates`-Feld mit dem `suburb`-Feld ersetzt, was vielmehr das Stadtviertel oder die Gemeinschaft des Users speichern soll. Der Ansatz, dass die `users`-Collection drei weitere Sub-Collections (`public`, `private` und `intern`) beinhalten soll wurde beibehalten.

Die `public`-Collection beinhaltet öffentliche Informationen von dem User, die für jeden anderen User zugänglich sind. Die öffentlichen Daten werden unter dem Dokument `data` gespeichert. Die Datenfelder der öffentlichen Daten haben sich auf die Folgenden angepasst:

- `birthday` für den Geburtstag des Nachbarn oder der Nachbarin
- `InTheNeighbourhoodSince` für das Datum seitdem der Nachbar oder die Nachbarin in der Nachbarschaft ist
- `Profession` für den Beruf des Nachbarn oder der Nachbarin
- `originallyFrom` für den Heimatsort des Nachbarn oder der Nachbarin
- `interests` für die Interessen des Nachbarn oder der Nachbarin
- `offers` für was angeboten werden kann von dem Nachbarn oder der Nachbarin
- `bio` für eine genauere Selbst-Beschreibung des Nachbarn oder der Nachbarin

Diese Daten werden nach wie vor vom User selbst festgelegt und dann beim öffentlichen Profil angezeigt. Der Grund für die Anpassungen ging dem Wunsch nach, das öffentliche Profil etwas simpler zu halten.

Die `private`-Collection beinhaltet private Informationen von dem User, zu denen kein anderer Zugriff hat. Zur besseren Aufteilung wurden die privaten Daten des Users auf mehrere Dokumente aufgeteilt.

Das Dokument `name` beinhaltet den Vornamen und Nachnamen, was —.

Als nächstes ist das Dokument `address`, welches die Addressdaten des Users beinhaltet, also das Feld `street` für die Straße, das Feld `streetNumber` für die Straßennummer, das Feld `city` für die Stadt und das Feld `zip` für die Postleitzahl.

Ein für den Nutzungswert der App wichtiges Dokument ist das Dokument `records`, welches die vom User markierten und geliketen Beiträge speichern soll. Das Dokument beinhaltet das Feld `posts`, welches die vom User markierten Posts speichert. Das Dokument beinhaltet außerdem zwei Sub-Collections: `likedPosts` und `likedComments`. Diese zwei Subcollections speichern die vom User geliketen Beiträge bzw. Kommentare. Die Funktionsweise der Subcollection `likedPosts` baut auf dem Konzept auf, dass die angelegten Entitäten nach der Id des Beitrags erstellt werden. Das heißt, dass die angelegte Entität die gleiche Id hat, wie der Beitrag, welcher vom User markiert wurde und sonst nichts. Das soll für eine möglichst effiziente Datenspeicherung sorgen. Die Funktionsweise der Subcollection `likedComments` baut auf dem selben Konzept auf, nur mit der Id des Kommentars als Entitäts-Id und zusätzlich wird noch die Id des Beitrags unter dem der Kommentar gelassen wurde mitgespeichert.

Das nächste Dokument `settings` soll die Benutzereinstellungen speichern. Darunter zählt das Feld `lastNameInitialOnly`, mit welchem der User bestimmt, ob sein ganzer Name angezeigt werden soll oder nur der Vorname mit dem Initial des Nachnamens, und `permReqBeforeChat`, mit welchem der User bestimmt, ob er/sie eine Anfrage erhalten möchte, bevor er/sie angeschrieben wird.

Das letzte Dokument `codes` soll die vom User erstellten Einlade-QR-Codes speichern. Um alle Einlade-QR-Codes vom User mittracken zu können, werden alle QR-Codes in einer Subcollection `generatedCodes` gesammelt. Der zuletzt generierte QR-Code und das Datum der Erstellung werden in die Felder `lastGeneratedCode` und `lastGeneratedCodeDate` gespeichert.

Die `intern`-Collection beinhaltet interne Informationen von dem User, die für die saubere Funktionalität der App wichtig sind. In dem neuen Datenmodell beinhaltet das Dokument zwei Dokemente: `address` und `verification`.

Im Dokument `address` werden im Feld `coords` die Koordinaten der angegeben Adresse gespeichert.

Falls sich der User durch einen Einlade-QR-Code verifiziert hat, hat das Dokument `verification` die Felder `deviceCoords` für die Gerätkoordinaten, `code` für den Einlade-QR-Code der eingesetzt wurde und `distance`, welches angibt wie groß die Distanz

zwischen den Gerätkoordinaten und den Koordinaten des Users, der den QR-Code erstellt hat, ist.

UserCoords

Die `userCoords`-Collection ist die Sammlung, die die Koordinaten des Users aufbewahrt. Die Dokumente der Collection beinhalten ein einziges Feld `g`. Dieses Feld ist eine Map, die aus den Feldern `geopoint` für die Koordinaten und `geohash` für den Geohash besteht. Der Geohash ist eine Kurzdarstellung der Koordinaten als alphanumerischer String. Die erneute Speicherung der User-Koordinaten als Hauptkollektion dient der Vereinfachung des Geofencing-Prozesses.

Codes

Die `codes`-Collection ist die Sammlung, wo alle QR-Codes die jemals erzeugt wurden, gespeichert ist. Die Dokumente der Collection beinhalten die Felder `coords` und `range` für die Koordinaten und Reichweite innerhalb der QR-Code gültig ist, `isActive` ob der Code noch gültig ist, `createdAt` für das Erstellungsdatum, `maxCodeLimit` für wie viele User mit diesem Einlade-QR-Code eingeladen werden können und `uid` für den User der den QR-Code erzeugt hat.

Posts

Die `posts`-Collection ist die Sammlung, die die Beiträge der User aufbewahren soll. Die Felder der Dokumente der Collection haben sich im Vergleich zum alten Datenmodell nicht verändert.

Was sich im Datenmodell im Bezug auf die Beiträge verändert hat, ist dass die `comments`-Collection als Subcollection zur `posts`-Collection hinzugefügt wurde. Die Felder der Dokumente der `comments`-Collection haben sich auch angepasst. Die Dokumente der Collection beinhalten die Felder `text` für den Text, `likes` für den Like-Zähler, `createdAt` für das Erstellungsdatum, `uid` für die Id des Users, welcher den Kommentar hinterlassen hat, und `postId` für die Id des Beitrags unter welchem der Kommentar hinterlassen wurde.

Notifications

Die `notification`-Collection ist die Sammlung, die die Benachrichtigungen der User aufbewahren soll. Benachrichtigungen sollen den User über neue Chat-Anfragen benachrichtigen. Die Dokumente der Collection beinhalten die Felder `fromUid` und `toUID` für

den User der die Benachrichtigung abgeschickt hat und den User der die Benachrichtigung erhält, `createdAt` für das Erstellungsdatum, `visible` ob die Benachrichtigung für den User sichtbar ist, `hasSeen` ob der User die Benachrichtigung gesehen hat und falls die Chat-Anfrage von einem Beitrag kommt noch das Feld `postId` für die Id des Beitrags. Außerdem existiert noch das Feld `type`, die den Typ der Anfrage beschreiben soll. Es gibt zwei Typen. Ein `postRequest` um für das Antworten eines Beitrags eine Chat-Anfrage zu erstellen und ein `userRequest` um zum Kontakt erstellen zu einen User eine Chat-Anfrage zu erstellen.

Reports

Die `reports`-Collection ist die Sammlung, wo die Meldungen auf Beiträge oder User gespeichert werden. Die Dokumente der Collection beinhalten die Felder `fromUser` für den User der die Meldung abgeschickt hat, `text` für die Erklärung warum die Meldung abgeschickt wurde, `createdAt` für das Erstellungsdatum, `visible` ob die Meldung für den User sichtbar ist und `reason` für den ausgewählten Grund warum die Meldung abgeschickt wurde. Außerdem existiert noch das Feld `type`, welches angibt, was gemeldet wurde. Es kann entweder ein Beitrag, Kommentar oder User gemeldet werden. Daneben gibt es noch das Feld `reportedId`, was die Id vom jeweiligen Beitrag, Kommentar oder User speichert.

Rooms

Die `rooms`-Collection ist die Sammlung, wo die Chats und dazugehörigen Nachrichten der User gespeichert werden. Die Dokumente der Collection beinhalten die Felder `userIds` für die User die im Chat verbunden werden, `createdAt` für das Erstellungsdatum und `type` für den Typ des Chats. Für den Typ eines Chats gibt es `direct` für einen Einzelchat zwischen zwei Usern und `group` für einen Gruppenchat zwischen mehreren Usern. Die App verwendet zum Zeitpunkt der Abgabe nur Einzelchats. Außerdem existieren noch die Felder `imageUrl` für das User-Foto und `name` für den User-Namen. Die zwei Felder werden während des Aufrufen des Chats mit den Daten des jeweils anderen User aktualisiert.

Damit User im Chat überhaupt schreiben können, benötigt es die Subcollection `messages`. Die Dokumente der Subcollection beinhalten die Felder `authorId` für den Verfasser der Nachricht, `text` für den verschickten Text, `createdAt` für das Erstellungsdatum und `type` für den Typ des Nachricht. Die grundlegendsten Typen einer Nachricht sind `text` für einen Text, `file` für eine Datei und `image` für ein Bild.

4.5.4 Cloud Storage

Der Cloud Storage dient in der Applikation als Speicherlösung für Bilder, einschließlich Profilbilder und Bilder, die zu Beiträgen hinzugefügt werden.

Aufbau

- profile
 - pictures
- posts
 - uid
 - * pictures

Die Dateistruktur teilt sich in zwei Ordner auf: `profile` und `posts`.

Im Ordner `profile` werden die Profilbilder der User gesichert. Dabei werden die Namen der Profilbilder als Id der User gesetzt, um das effizientere Speichern und Suchen von Profilbildern einzelner User zu erreichen.

Im Ordner `posts` werden die Bilder die zu Beiträge gehören gesichert. Dabei besitzt der `posts`-Ordner noch weitere Unterordner die nach der Id der User benannt werden. Das soll garantieren, dass die Bilder, die von User hochgeladen werden, nur in deren Ordner gespeichert werden.

Funktionsweise

Listing 11: uploadProfileImageToStorage Funktion

```

1   Future<String> uploadProfileImageToStorage(Uint8List file) async {
2     final authService = Get.find<AuthService>();
3
4     final uid = authService.uid;
5
6     if (uid.isEmpty) {
7       return '';
8     }
9
10    Reference ref = FirebaseStorage.instance.ref().child('profile/$uid');
11    UploadTask uploadTask = ref.putData(file);
12
13    TaskSnapshot snap = await uploadTask;
14    String downloadUrl = await snap.ref.getDownloadURL();
15    return downloadUrl;
16  }

```

Der vorliegende Code 11 soll die Funktionsweise des Cloud Storage demonstrieren. Die Funktion soll das Profilbild des Users im Cloud Storage einspeichern. Im Code lässt sich sehen, dass die Funktion das Bild in Form einer `Uint8List` bekommt und sich zuerst die Id des aktuellen Users durch den `AuthService` holt. Danach wird eine Referenz zum Cloud Storage auf den Pfad `profile/$uid` gebildet und die Bilddatei wird auf die Referenz hochgeladen. Wenn das Hochladen erfolgreich war, wird die URL, unter der das Bild verfügbar ist, zurückgegeben. Diese URL kann dann in den Userdaten unter der `users`-Collection im Feld `imageUrl` gespeichert werden.

4.5.5 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.

Alle Cloud Functions wurden in TypeScript entwickelt, da diese Programmiersprache Typsicherheit und verbesserte Fehlererkennung bietet.

Überprüfen der Adresse

Die Cloud-Funktion `checkAddress` erfordert eine authentifizierte Anfrage und erhält eine Adresse vom User, die aus Straße, Hausnummer, Stadt und Postleitzahl besteht. Um sicherzustellen, dass die Anfrage authentifiziert ist, wird zuerst überprüft, ob der Aufrufer authentifiziert ist. Wenn das nicht der Fall sein sollte, wird eine Fehlermeldung ausgegeben, die besagt, dass die Anfrage nicht authentifiziert ist.

Die Funktion überprüft danach, ob alle erforderlichen Daten für die Adresse vorhanden sind. Wenn Daten fehlen, wird eine Fehlermeldung ausgegeben, die besagt, dass gültige

Adressdaten benötigt werden.

Die Adresse wird dann formatiert und an die Funktion `getOSMCoordinatesFromAddress` übergeben, um die Koordinaten der Adresse abzurufen. Wenn ein Fehler auftritt, was passieren kann, wenn eine nicht existierende Adresse angegeben wird, wird eine Fehlermeldung ausgegeben, die besagt, dass die Koordinaten nicht abgerufen werden konnten. Falls die Adresse verifiziert werden konnte, werden die Addressdaten vom User, also Straße, Hausnummer, Stadt und Postleitzahl, im Firestore gespeichert.

Am Ende gibt die Funktion `true` zurück, um anzugeben, dass die Verifizierung erfolgreich abgeschlossen wurde.

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 anzugeben, dass der Benutzer erfolgreich verifiziert wurde. Die Cloud-Funktion gibt `true` zurück, um anzugeben, dass die Verifizierung erfolgreich war.

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

Registrierung 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

Für die Verifizierung ist es von Bedeutung, die Adresse des Nutzers in Koordinaten umzuwandeln, um im Backend damit arbeiten zu können. Ursprünglich wurde die Google Geocode API verwendet, jedoch ist diese kostenpflichtig. Eine passendere Alternative, die besser zur Projektphilosophie passt, wurde gefunden: Nominatim. Nominatim ist eine Open-Source-Geocoding-API für OpenStreetMap-Daten und bietet eine kostenlose API für diese spezielle Anforderung.

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 12: `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     return d;
31   }
32
33   function deg2rad(deg: number) {
34     return deg * (Math.PI / 180);
35   }

```

Der vorliegende Code 12 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 [72] 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 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.

Bestimmung der nächstgelegenen Entfernung

Zur Wahrung der Privatsphäre der Nachbarn ist eine Funktion erforderlich, die den nächstgelegenen Abstand zu den Nachbarn angibt, ohne den genauen Abstand offenzulegen. Diese Funktion trägt den Namen `getNearestDistance` und erhält 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.

Letztendlich wird geprüft, ob `nearest` größer oder gleich 1000 ist. Abhängig davon wird der Abstand entweder in Kilometern oder Metern zurückgegeben. Ist der Abstand größer oder gleich 1000, wird die Einheit `km` hinzugefügt, andernfalls wird `m` verwendet.

Verzichtet wurde bewusst auf Switches oder If-Bedingungen, da mit der aktuellen Funktion schneller Abstände geändert werden können. Die Evaluierung des Arrays mit den Optionen ist noch im Gange.

Nachbarschaft von Nutzer bestimmen

Um den App-Nutzern ein besseres Verständnis der Nachbarschaften zu vermitteln, in denen ihre Nachbarn leben, wird bei jedem Benutzer die jeweilige Nachbarschaft angezeigt. Diese Information wird automatisch in der Datenbank gespeichert, sobald der Nutzer verifiziert ist. Zur Umsetzung dieser Funktion wird eine Methode eingesetzt, 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 `String` 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.

Beiträge von Nachbarn in der Umgebung ermitteln

Die Cloud Function `getUserPostsWithinRange` ist dafür verantwortlich, Beiträge von User innerhalb einer bestimmten Reichweite abzurufen. Dies wird durch den Aufruf von Firestore und GeoFireStore erreicht, um die User innerhalb des Bereichs zu ermitteln und dann ihre Beiträge abzurufen.

GeoFirestore[73] ist eine Open-Source-Bibliothek, die auf der Firebase-Plattform basiert und die Funktionen von Firestore erweitert, um Geofencing und so die Filterung von Daten nach geografischen Entfernung und Koordinaten zu ermöglichen. Quelle: <https://geofirestore.com>

Die Cloud Function bekommt als Parameter die Reichweite innerhalb der gefiltert werden soll und prüft zunächst, ob der User authentifiziert ist. Wenn nicht, wird ein Fehler ausgelöst, der besagt, dass die Anfrage nicht authentifiziert ist. Wenn der User authentifiziert ist, wird seine Id verwendet, um seine Koordinaten aus der Firestore-Datenbank abzurufen.

Die Funktion verwendet dann GeoFireStore, um alle User innerhalb der mitgegebenen Reichweite zu ermitteln mit den Userkoordinaten als Mittelpunkt. Diese emittelten User werden dann gefiltert, um deren Beiträge aus der Firestore-Datenbank abzurufen. Um sicherzustellen, dass der User innerhalb der Reichweite in der, der Beitrag sichtbar sein soll, liegt, wird für jeden Beitrag die Entfernung zwischen dem User, der die Cloud Functions aufgerufen hat, und dem User, der den Beitrag erstellt hat, berechnet. Wenn die Entfernung kleiner als die Reichweite innerhalb der Beitrag sichtbar sein soll ist, wird der Beitrag in die Liste der gültigen Beiträge aufgenommen.

Schließlich gibt die Funktion eine Liste mit den Ids der gültigen Beiträge zurück, die innerhalb der mitgegebenen Reichweite liegen.

Die Grafik 46 zeigt die Funktionsweise der Cloud Function beispielsweise als Bild an.

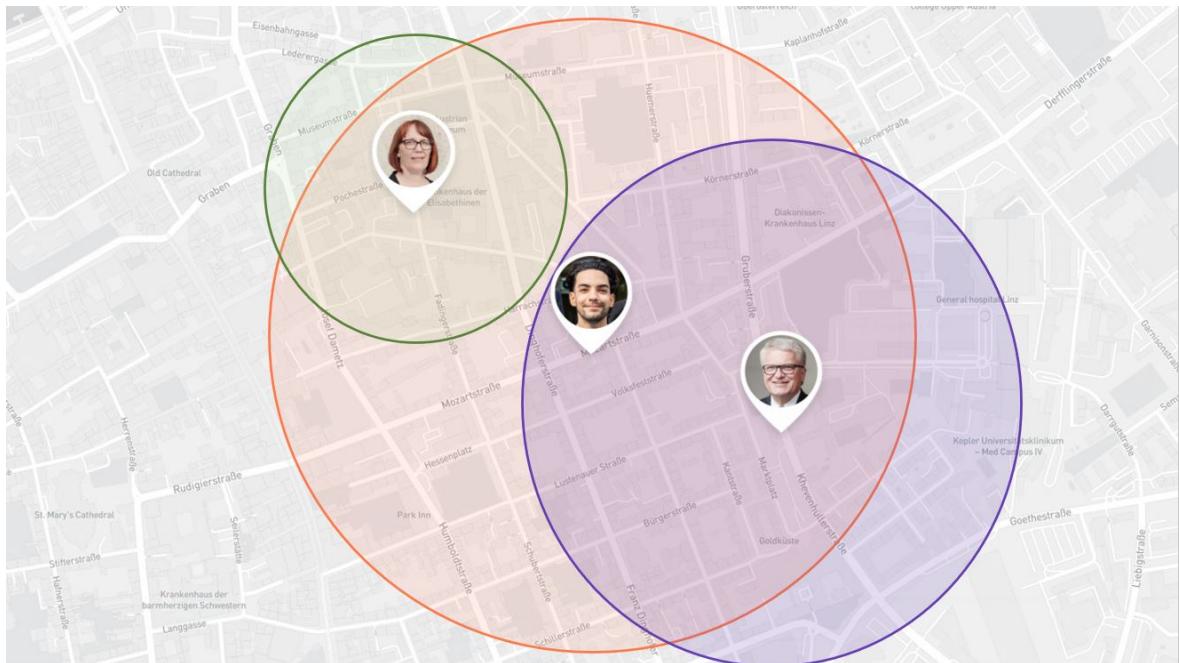


Abbildung 46: Reichweiten verschiedener Nachbarn

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 Verifizie-

rungscode 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.

Likes von Beiträge steuern

Die Verwendung einer Cloud-Function zur Änderung des Like-Zählers erfolgt aus dem Grund, dass Daten von einem Beitrag in der Regel nur vom Ersteller des Beitrags verändert werden können. Um dennoch sicherzustellen, dass alle Nutzer, die den Beitrag mögen, den Like-Zähler erhöhen können, wird eine Cloud-Function verwendet, die automatisch auf die Hinzufügung von Likes reagiert und den Zähler aktualisiert. Dadurch

wird sichergestellt, dass der Like-Zähler korrekt und in Echtzeit aktualisiert wird, ohne dass der Ersteller des Beitrags manuell Änderungen vornehmen muss.

Likes von Beiträge erhöhen

Listing 13: incrementLikeCounterOfPost Funktion

```

1  export const incrementLikeCounterOfPost = functions
2  .region("europe-west1")
3  .firestore.document("users/{userId}/private/record/likedPosts/{likedPostId}")
4  .onCreate(async (snapshot, context) => {
5    const likedPostId = context.params.likedPostId;
6
7    const likedPost = db.collection("posts").doc(likedPostId);
8
9    await db.runTransaction(async (transaction) => {
10      const snapshot = await transaction.get(likedPost);
11      const newLikeCounter = snapshot.get("likes") + 1;
12      transaction.update(likedPost, { likes: newLikeCounter });
13    });
14  });

```

Der vorliegende Code 13 implementiert einen Cloud-Function-Trigger, der es ermöglicht, Funktionen automatisch auszulösen, wenn bestimmte Ereignisse auftreten. Die Funktion verwendet dabei Firebase Cloud Functions, um auf Änderungen in der Datenbank zu reagieren und entsprechende Aktionen auszuführen.

Die Cloud-Funktion `incrementLikeCounterOfPost` wird aufgerufen, wenn ein neuer Eintrag in der Firestore-Sammlung `users/userId/private/record/likedPosts` erstellt wird. Das Ziel dieser Funktion besteht darin, den Like-Zähler eines Beitrags in der Firestore-Sammlung `posts` zu erhöhen, wenn ein Nutzer den Beitrag als *gefällt mir* markiert.

Der erste Schritt der Funktion besteht darin, die ID des Beitrags zu ermitteln, dessen Like-Zähler erhöht werden soll. Dies geschieht durch den Zugriff auf den Kontext-Parameter `likedPostId`. Anschließend wird eine Referenz auf den entsprechenden Beitrag in der Firestore-Sammlung `posts` erstellt.

In einem nächsten Schritt wird eine Transaktion gestartet, um sicherzustellen, dass die Like-Zähler-Operation atomar ausgeführt wird. In dieser Transaktion wird der aktuelle Like-Zähler des Beitrags abgerufen und um eins erhöht. Die aktualisierte Anzahl wird dann in der Firestore-Sammlung `posts` gespeichert.

Die Verwendung von Transaktionen gewährleistet, dass die Like-Zähler-Operation atomar ausgeführt wird und es somit zu keiner Inkonsistenz der Daten kommt. Wenn mehrere Nutzer gleichzeitig den gleichen Beitrag als *gefällt mir* markieren würden, könnte es sonst zu Konflikten und Fehlern bei der Aktualisierung des Like-Zählers kommen.

Likes von Beiträge verringern

Listing 14: decrementLikeCounterOfPost Funktion

```

1  export const decrementLikeCounterOfPost = functions
2    .region("europe-west1")
3    .firestore.document("users/{userId}/private/record/likedPosts/{likedPostId}")
4    .onDelete(async (snapshot, context) => {
5      const likedPostId = context.params.likedPostId;
6
7      const likedPost = db.collection("posts").doc(likedPostId);
8
9      await db.runTransaction(async (transaction) => {
10        const snapshot = await transaction.get(likedPost);
11        const newLikeCounter = snapshot.get("likes") - 1;
12        transaction.update(likedPost, { likes: newLikeCounter });
13      });
14    });

```

Der vorliegende Code 14 implementiert einen Cloud-Function-Trigger, der ähnlich zum vorherigen Trigger aufgebaut ist.

Die Cloud-Funktion `incrementLikeCounterOfPost` wird aufgerufen, wenn ein Eintrag in der Firestore-Sammlung `users/userId/private/record/likedPosts` gelöscht wird. Der Zweck dieser Funktion besteht darin, den Like-Zähler des Beitrags in der Firestore-Sammlung `posts` zu verringern, wenn ein Nutzer seinen Like entfernt.

Auch hier wird eine Transaktion verwendet, um sicherzustellen, dass die Änderung atomar ausgeführt wird und somit keine Inkonsistenzen in den Daten entstehen. Die Funktion ruft dabei den aktuellen Like-Zähler des Beitrags ab und verringert diesen um eins. Die aktualisierte Anzahl wird dann in der Firestore-Sammlung `posts` gespeichert. Die Verwendung von Cloud-Funktionen zur Verwaltung der Like-Zähler stellt sicher, dass diese in Echtzeit aktualisiert werden und Nutzer eine korrekte und aktuelle Anzahl der Likes sehen.

Likes von Kommentare steuern

Ähnlich wie bei der Verwaltung der Like-Zähler von Beiträgen werden auch bei der Verwaltung von Likes für Kommentare Cloud-Funktionen eingesetzt. Diese gewährleisten, dass die Like-Zähler in Echtzeit aktualisiert werden und Nutzer eine korrekte und aktuelle Anzahl der Likes sehen können.

Likes von Kommentare erhöhen

Listing 15: incrementLikeCounterOfComment Funktion

```

1  export const incrementLikeCounterOfComment = functions

```

```

2   .region("europe-west1")
3   .firestore.document("users/{userId}/private/record/likedComments/{likedCommentId}")
4   .onCreate(async (snapshot, context) => {
5     const likedCommentId = context.params.likedCommentId;
6     const likedCommentRef = snapshot.ref;
7
8     const likedComment =
9       db.collection("posts").doc(likedCommentRef['post']).collection('comments')
10      .doc(likedCommentId);
11
12     await db.runTransaction(async (transaction) => {
13       const snapshot = await transaction.get(likedComment);
14       const newLikeCounter = snapshot.get("likes") + 1;
15       transaction.update(likedComment, { likes: newLikeCounter });
16     });
17   });

```

Der vorliegende Code 15 implementiert einen weiteren Cloud-Function-Trigger.

Die Cloud-Funktion `incrementLikeCounterOfComment` wird aufgerufen, wenn ein neuer Eintrag in der Firestore-Sammlung `users/userId/private/record/likedComments` erstellt wird. Das Ziel dieser Funktion besteht darin, den Like-Zähler eines Kommentars in der Firestore-Sammlung `posts` zu erhöhen, wenn ein Nutzer den Kommentar als *gefällt mir* markiert.

Der erste Schritt der Funktion besteht darin, die ID des Kommentars zu ermitteln, dessen Like-Zähler erhöht werden soll. Dies geschieht durch den Zugriff auf den Kontext-Parameter `likedCommentId`. Anschließend wird eine Referenz auf den entsprechenden Kommentar in der Firestore-Sammlung `posts` erstellt.

In einem nächsten Schritt wird eine Transaktion gestartet, um sicherzustellen, dass die Like-Zähler-Operation atomar ausgeführt wird. In dieser Transaktion wird der aktuelle Like-Zähler des Kommentars abgerufen und um eins erhöht. Die aktualisierte Anzahl wird dann in der Firestore-Sammlung `posts` gespeichert.

Likes von Kommentare verringern

Listing 16: `decrementLikeCounterOfComment` Funktion

```

1   export const decrementLikeCounterOfComment = functions
2   .region("europe-west1")
3   .firestore.document("users/{userId}/private/record/likedComments/{likedCommentId}")
4   .onDelete(async (snapshot, context) => {
5     const likedCommentId = context.params.likedCommentId;
6     const likedCommentRef = snapshot.ref;
7
8     const likedComment =
9       db.collection("posts").doc(likedCommentRef['post']).collection('comments')
10      .doc(likedCommentId);
11
12     await db.runTransaction(async (transaction) => {
13       const snapshot = await transaction.get(likedComment);
14       const newLikeCounter = snapshot.get("likes") - 1;
15       transaction.update(likedComment, { likes: newLikeCounter });
16     });
17   });

```

Der vorliegende Code 16 die Cloud-Funktion `decrementLikeCounterOfComment`, welche aufgerufen wird, wenn ein Eintrag in der Firestore-Sammlung `users/userId/private/record/like` gelöscht wird. Ziel dieser Funktion ist es, den Like-Zähler des Kommentars in der Firestore-Sammlung `posts` zu verringern, wenn ein Nutzer seinen Like entfernt. Ähnlich zur `incrementLikeCounterOfComment`-Funktion ermittelt die Funktion zunächst die ID des entsprechenden Kommentars und erstellt eine Referenz auf diesen in der Firestore-Sammlung `posts`. Danach wird eine Transaktion gestartet, um sicherzustellen, dass die Like-Zähler-Operation atomar ausgeführt wird. Der aktuelle Like-Zähler des Kommentars wird abgerufen und um eins verringert. Die aktualisierte Anzahl wird dann in der Firestore-Sammlung `posts` gespeichert.

Typesense Search

Das Nochba-Team versuchte ursprünglich, Typesense für die Suchfunktionalität zu verwenden, da es sich um eine schnelle, fehlertolerante Suchmaschine handelt, die für sofortige Suchvorgänge optimiert ist. Aufgrund der unvollständigen Dokumentation hatte das Team jedoch Schwierigkeiten, Typesense in die Flutter-Applikation zu implementieren. Das Team versuchte, die Typesense-Entwickler um Unterstützung zu bitten, aber dieser Schritt war sehr zeitaufwändig. Das Team entschied sich daher, Algolia als Alternative zu Typesense zu verwenden.

4.5.6 Algolia Search

Die finale Entscheidung des Nochba-Teams fiel auf die Algolia-Suche, da sowohl Typesense als auch Algolia den gleichen Feature-Set für Suchfunktionen bieten. Ein bemerkenswerter Unterschied zwischen beiden ist, dass Algolia eine umfangreichere Dokumentation und Unterstützung bietet, was es für Entwickler einfacher macht, die Suchfunktion zu implementieren und in ihre Anwendung zu integrieren.

4.6 Verwaltungslogik

Abbildung 47 stellt die Struktur dar, die es ermöglicht Daten von dem Firestore-Server abzurufen und in der Flutter-App zu verwalten. Die Struktur besteht aus mehreren Komponenten, die im Folgenden näher erläutert werden:

Die Struktur beginnt auf der UI-Ebene mit der Seite, die Daten aus dem Firestore benötigt. Diese Seite verfügt über einen GetxController, der für die Verwaltung des

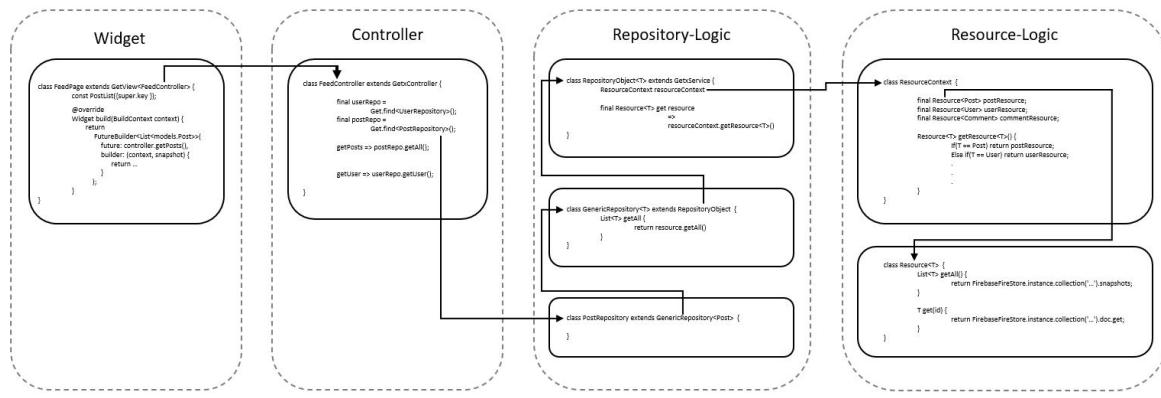


Abbildung 47: Verwaltungslogik Diagramm

Zustands und die Implementierung von Logik zuständig ist. Dieser Controller initialisiert Repositories, die den Zugriff auf die Daten vereinfachen.

Jedes Repository erbt von einer GenericRepository-Klasse. Das GenericRepository ist ein generischer Typ, der es ermöglicht, Repositories für verschiedene Datenmodelle zu erstellen, ohne dass jeder Repository-Typ dieselben grundlegenden Operationen implementieren muss. Das bedeutet, dass ein Repository für jede Sammlung im Firestore erstellt werden kann, indem einfach der entsprechende Typ als generischer Parameter angegeben wird.

Ein wichtiger Vorteil bei der Verwendung von generischen Repositories besteht darin, dass dadurch Code-Duplikation vermieden wird. Durch die Implementierung von grundlegenden CRUD-Operationen in der GenericRepository-Klasse müssen diese nicht in jedem Repository wiederholt werden, was die Wartbarkeit des Codes erhöht und die Wahrscheinlichkeit von Fehlern verringert.

Darüber hinaus ermöglicht die Verwendung von Generics auch die einfache Erweiterung und Anpassung der Implementierung. Wenn beispielsweise spezielle Anforderungen für bestimmte Datenmodelle bestehen, können diese problemlos in einem spezialisierten Repository hinzugefügt werden, ohne dass andere Teile des Codes beeinflusst werden. Die GenericRepository-Klasse erbt von der RepositoryObject-Klasse, welche den ResourceContext beinhaltet. Der ResourceContext stellt den Zugriff auf unterschiedliche Ressourcen bereit, wodurch die Interaktion mit den zugrunde liegenden Datenquellen ermöglicht wird.

Jede Ressource ist für den Zugriff auf Daten einer entsprechenden Sammlung innerhalb des Firestore zuständig und implementiert grundlegende CRUD-Operationen. Durch die Kapselung dieser Operationen in dedizierten Ressourcen wird eine klare Trennung der Verantwortlichkeiten erreicht.

Darüber hinaus erlaubt die Verwendung von ResourceContext eine einheitliche Schnitt-

stelle für den Zugriff auf verschiedene Ressourcen, wodurch die Interoperabilität zwischen verschiedenen Komponenten der Verwaltungslogik gefördert wird. Diese Architektur erlaubt es, neue Ressourcen hinzuzufügen oder bestehende Ressourcen anzupassen, ohne die Gesamtstruktur der Anwendung zu beeinträchtigen.

4.7 Flutter Packages

Im Folgenden finden Sie eine Auswahl an essentiellen und nützlichen Flutter-Pakete, die in der App zum Einsatz kam.

firebase_core: 2.0.0[74] - Dieses Paket bildet die grundlegende Firebase-Bibliothek, die für die Verwendung anderer Firebase-Pakete in der Applikation erforderlich ist. Initialisiert Firebase und stellt die Konfiguration bereit.

firebase_auth: 4.0.1[75] - Dieses Paket erlaubt die Integration der Firebase-Authentifizierung in der Applikation. Es unterstützt die Anmeldung mit verschiedenen Anbietern, wie beispielsweise E-Mail oder Google.

cloud_firestore: 4.0.1[76] - Dieses Paket erlaubt die Integration von Firestore in der Applikation. Es unterstützt das Erstellen, Bearbeiten und Löschen von Daten in Echtzeit.

firebase_storage: 11.0.1[77] - Dieses Paket erlaubt die Integration von Cloud Storage in der Applikation. Es ermöglicht das Hoch- und Herunterladen von Dateien und Bildern.

cloud_functions: 4.0.6[78] - Dieses Paket erlaubt die Verwendung von Cloud Functions in der Applikation. Wird für serverlose Logik und Backend-Integration verwendet.

get: 4.6.5[44] - Dieses Paket erlaubt die Integration von GetX in der Applikation. Es unterstützt das App-Management, die Navigation, die Zustandsverwaltung und Abhängigkeitsinjektionen.

openai_client: 0.0.7[79] - Dieses Paket ermöglicht die Nutzung von OpenAI-Diensten in der Applikation, wodurch die Integration von künstlicher Intelligenz und maschinellem Lernen unterstützt wird.

algolia: 1.1.1[80] - Dieses Paket ermöglicht die Integration der Algolia-Suchplattform in der Applikation und bietet schnelle, relevante Suchergebnisse in Echtzeit.

flutter_map: 3.1.0[81] - Dieses Paket ermöglicht die Integration von interaktiven Karten in der Applikation.

google_mlkit_translation: 0.5.0[82] - Dieses Paket ermöglicht die Integration der Google ML Kit Translation-Funktion in der Applikation. Es unterstützt die Textüber-

setzungen in Echtzeit.

google_mlkit_language_id: 0.5.0[83] - Dieses Paket ermöglicht die Integration der Googles ML Kit Language Identification in der Applikation. Es unterstützt die Identifizierung von Sprachen in Texten.

google_sign_in: 6.0.2[84] - Dieses Paket ermöglicht die Integration von Google Konten in der Applikation. Unterstützt die Google-Anmeldung.

feedback: 2.5.0[85] - Dieses Paket ermöglicht das sammeln von Benutzerfeedback mit Screenshots und Anmerkungen, die an das Entwicklungsteam gesendet werden können.

mobile_scanner: 3.0.0[86] - Dieses Paket ermöglicht die Integration von Scanner-Funktionalitäten für mobile Geräte in der Applikation und unterstützt das Scannen von Barcodes, QR-Codes und Dokumenten.

http: 0.13.4[87] - Dieses Paket ermöglicht die Verwendung von HTTP-Anfragen in der Applikation. Es unterstützt GET, POST, PUT, DELETE und andere HTTP-Methoden.

syncfusion_flutter_datepicker: 20.2.45[88] - Dieses Paket ermöglicht die Verwendung von benutzerfreundlichen und anpassbaren Datumsauswahl-Funktionalitäten in der Applikation.

image: 4.0.15[89] - Dieses Paket ermöglicht in der Applikation die Dekodierung, Bearbeitung und Kodierung von Bildern in verschiedenen Formaten wie PNG, JPEG, GIF und mehr.

image_picker: 0.8.5+3[90] - Dieses Paket ermöglicht in der Applikation das Auswählen von Bildern aus der Galerie oder das Aufnehmen neuer Bilder mit der Kamera.

image_editor_plus: 0.2.0[91] - Dieses Paket ermöglicht in der Applikation eine einfach zu bedienende Bildbearbeitungs-Ansicht, das grundlegende Bearbeitungsfunktionen wie Zuschneiden, Drehen und Skalieren beherrscht.

cupertino_icons: 1.0.5[92] - Dieses Paket stellt eine Sammlung von über 1.000 hochwertigen iOS-Stil-Icons (Cupertino) für die Verwendung in der Applikation zur Verfügung.

get_time_ago: 1.1.6:[93] - Dieses Paket ermöglicht in der Applikation das Umwandeln von Datumstypen in menschenlesbare Zeitangaben mit vordefinierbaren Layout.

4.8 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.

4.8.1 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.

4.8.2 Hosting-Anbieter

Um den Aufwand zu minimieren, wurde der Hosting-Anbieter [94] 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.

4.8.3 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.

4.8.4 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 Edalatkhan genutzt wurde.

4.8.5 Vorteile

Rückblickend erwies sich die Entscheidung, eine eigene E-Mail-Adresse zu erstellen, als äußerst vorteilhaft. Sie vermittelte einen professionellen Eindruck und führte dazu, dass das Projekt ernster genommen wurde. Zudem erleichterte die eigene E-Mail-Adresse die Verwaltung von E-Mails an einem zentralen Ort. Dies war besonders praktisch, da im

Laufe des Projekts verschiedene Anfragen eingingen und so die Kommunikation effizienter gestaltet werden konnte.

4.9 Visual Studio Code Extention

Für die Programmierung einer App wurde das Werkzeug ChatGPT eingesetzt, das auch nach der Veröffentlichung von GPT-4 relevanter wurde. Eine einfache VSCode-Erweiterung wurde entwickelt, um den Dart-Code zu minimieren. Da GPT-4 jedoch nur eine Eingabe von 8000 Tokens zulässt, ist es schnell erreicht, wenn der Code nicht minimiert wird, insbesondere wenn viele Leerzeichen aufgrund von Einrückungen vorhanden sind. Jedes Leerzeichen stellt dabei ein Token dar, was schnell zu einer Überschreitung der maximalen Tokenanzahl führen kann. Die Erweiterung wurde vollständig von GPT-4 programmiert und ist unter [95] verfügbar.

Später wurde eine weitere Erweiterung programmiert, die die Anzahl der Tokens im aktuellen Code-File mithilfe von Codex oder GPT-3 Tokenizer anzeigt. Wenn ein bestimmter Bereich im Code-File ausgewählt ist, wird nur die Anzahl der Tokens in diesem Bereich angezeigt. Die Erweiterung wurde im VSCode Marketplace veröffentlicht und ist unter [96] erhältlich.

5 Evaluieren getroffener Entscheidungen

5.1 Zukünftige mögliche Implementierungen

Das Ziel des Teams besteht darin, die App jedem Österreicher und jeder Österreicherin zugänglich zu machen. Daher wurden folgende Ideen während des Brainstormings entwickelt:

5.1.1 Benutzererfahrung (UX) und Benutzeroberfläche (UI)

- **UI/UX-Überarbeitung:** Optimierung der Benutzeroberfläche und Benutzererfahrung für eine intuitivere und ansprechendere App-Nutzung.
- **Vereinfachung der Beitragserstellung:** Einfacheres Erstellen von Beiträgen durch verbesserte Benutzerführung.
- **Automatische Kategorisierung von Beiträgen:** Einsatz von KI, um Beiträge automatisch passenden Kategorien zuzuordnen.
- **Anpassbare Benachrichtigungseinstellungen:** Benutzer können individuelle Einstellungen für Benachrichtigungen festlegen.

5.1.2 Registrierung und Profil

- **Vereinfachung des Registrierungsprozesses:** Adressen-Autocomplete-Funktion, um den Registrierungsprozess schneller und einfacher zu gestalten.
- **Identitätsverifizierung:** Optionale Verifizierung der Benutzeridentität anhand eines Ausweises, um Vertrauen innerhalb der Community aufzubauen.

5.1.3 Kommunikation und Interaktion

- **Neuentwicklung des Chats:** Einführung von Gruppenchats, Chat-Reaktionen, Sprachnachrichten und erweiterten Anhangsoptionen.

- **Punktesystem:** Belohnung von aktiven Nachbarn durch ein Karma-ähnliches Punktesystem.
- **Integration von Veranstaltungskalendern:** Hinzufügen von lokalen Events und Veranstaltungskalendern zur App, um die Nachbarschaftsinteraktion zu fördern.

5.1.4 Hilfe und Support

- **Erweiterte Hilfe und Tutorials:** Bereitstellung von umfassenden Hilfeinhalten und Tutorials, um Benutzern den Einstieg zu erleichtern.

5.1.5 Technische Verbesserungen

- **Umfangreiches Caching:** Implementierung von Caching-Strategien, um die App-Performance zu verbessern und die Ladezeiten zu reduzieren.
- **Chat-Verschlüsselung:** Einführung einer Ende-zu-Ende-Verschlüsselung für Chats, um die Privatsphäre und Sicherheit der Benutzer zu gewährleisten.
- **Deep Links:** Ermöglichen von Deep Links für jede Seite, um das Teilen von Inhalten und die Benutzerfreundlichkeit zu erhöhen.

5.1.6 Sicherheit und Datenschutz

- **Erweiterte Sicherheitsmechanismen:** Implementierung zusätzlicher Sicherheitsfunktionen, um die App vor Bedrohungen zu schützen.
- **Erweiterte Benachrichtigungsoptionen:** Verbesserung der Benachrichtigungsoptionen, um Benutzer über wichtige Sicherheits- und Datenschutzupdates zu informieren.

5.1.7 Meilensteine

Zu Beginn der Diplomarbeit wurden folgenden Meilensteine festgelegt:

10.07.2022	Ein Interface-Prototyp ist designt
17.07.2022	Die Systemarchitektur ist definiert und die Machbarkeit geprüft
07.08.2022	Ein Minimum Viable Product, welches die Funktionen Login, Feed, Post erstellen, Filter und Chat-Funktion beinhaltet, ist implementiert
07.01.2023	Ein Minimum Loveable Product, welches die Funktionen Registrierung, Suche, Kontoeinstellungen, Benachrichtigungs-Tab, Übersetzung, Kommentarfunktion, mehrere Kategorien und Report-System beinhaltet, ist implementiert
26.02.2023	Alle geplanten Funktionen sind vollständig implementiert
27.03.2023	Unvollkommenheiten sind durch Kunden-Feedback und Bugfixing behoben
01.04.2023	Diplomarbeit ist vollständig abgeschlossen und abgeben

Unter den Meilensteinen wurden die wichtigsten Etappen des Projekts definiert. Allerdings konnten die festgelegten Termine, wegen Überschätzung des Arbeitsaufwandes, nicht eingehalten werden.

Trotz der Verzögerungen konnten jedoch das Minimum Viable Product und das Minimum Loveable Product erfolgreich fertiggestellt werden.

In den Wochen vor der Abgabe wurden noch Bugs gesucht und behoben, um die Qualität der App weiter zu verbessern.

Da die App erst sehr spät auf dem Playstore veröffentlicht wurde, ist es dem Team zum Zeitpunkt der Abgabe nicht gelungen, Kundenfeedback einzuholen.

5.2 Erfahrungen

Während der Erstellung der Diplomarbeit hat das Team wertvolle Erfahrungen gesammelt und umfangreiches Wissen in verschiedenen Technologien erworben. Die Erfahrungen lassen sich in verschiedene Bereiche aufteilen:

5.2.1 Frontend-Entwicklung mit Flutter

Die Diplomarbeit bot dem Team die Möglichkeit, sich intensiv mit der Frontend-Entwicklung auseinanderzusetzen und dabei die Vorteile von Flutter als Framework zu erkennen. Da die meisten Teammitglieder bis dato keine oder nur wenig Erfahrung in diesem Bereich hatten, stellte dies eine wertvolle Lernerfahrung dar.

Die Teammitglieder eigneten sich Kenntnisse über den grundsätzlichen Aufbau von Flutter-Anwendungen an, wie zum Beispiel die Verwendung von Widgets, State Management und das Implementieren von Animationen. Die Herausforderung bestand darin, die Funktionalität der Anwendung in einer strukturierten und effizienten Weise zu entwickeln, während gleichzeitig ein ansprechendes Design und Benutzeroberfläche beibehalten wurde.

5.2.2 Backend-as-a-Service mit Firestore

Das Team konnte wertvolle Erfahrungen im Umgang mit einer BaaS wie Firestore sammeln.

Die Teammitglieder lernten, wie sie Firestore-Dokumente und -Sammlungen erstellen, abfragen und manipulieren können, um die erforderlichen Daten für die Anwendung bereitzustellen. Außerdem erwarb das Team Kenntnisse darüber, wie Security Rules effektiv formuliert und angewendet werden können.

Darüber hinaus ermöglichte das Arbeiten mit Cloud Storage den Teammitgliedern, Erfahrungen im Umgang mit Dateiuploads und -downloads zu sammeln, während Cloud Functions dazu beitrugen, serverseitige Funktionen für die Anwendung besser zu verstehen.

5.2.3 Teamarbeit außerhalb der Schule

Die Diplomarbeit stellte für das Team auch eine Gelegenheit dar, Teamarbeit außerhalb des schulischen Rahmens zu erfahren. Die Teammitglieder lernten, wie sie effektiv zusam-

menarbeiten, Kommunikationskanäle einrichten und Ressourcen verwalten können, um ein erfolgreiches Projekt abzuschließen. Die Zusammenarbeit erforderte die Entwicklung von Soft Skills wie Zeitmanagement, Planung und Organisation, sowie die Fähigkeit, Feedback anzunehmen und konstruktive Kritik zu üben.

Die Teammitglieder nutzten verschiedene Projektmanagement-Tools und Kommunikationsplattformen, um den Arbeitsfortschritt zu verfolgen, Aufgaben zuzuweisen und Probleme gemeinsam zu lösen. Dies führte zu einer verbesserten Effizienz und half dabei, den Fokus auf die wichtigsten Aspekte des Projekts zu legen.

Zudem lernten die Teammitglieder, wie sie ihre individuellen Stärken und Fähigkeiten am besten einsetzen können, um das Projekt voranzubringen. Durch die Verteilung der Verantwortlichkeiten und die Arbeitsteilung konnte das Team sicherstellen, dass jeder Beitrag zum Gesamterfolg des Projekts beitrug.

5.2.4 Teilnahme an Wettbewerben

Die Erfahrungen, die das Team während der Diplomarbeit gesammelt hat, wurden auch durch die Teilnahme an Wettbewerben erweitert und gefestigt. Die Teilnahme an solchen Veranstaltungen ermöglichte den Teammitgliedern, ihr Wissen und ihre Fähigkeiten in einem wettbewerbsorientierten Umfeld zu testen und ihre Arbeit gegenüber anderen Projekten zu validieren.

Die Teilnahme an Wettbewerben bot dem Team die Möglichkeit, von der Expertise der Jury und anderen Teilnehmern zu profitieren und wertvolles Feedback für die Weiterentwicklung und Verbesserung des Projekts zu erhalten.

Des Weiteren trug die Teilnahme an Wettbewerben zur Motivation und zum Zusammenhalt des Teams bei. Die Anerkennung durch Fachleute und die Möglichkeit, Preise und Auszeichnungen zu gewinnen, spornten die Teammitglieder an, ihr Bestes zu geben und kontinuierlich an der Optimierung des Projekts zu arbeiten.

Literaturverzeichnis

- [1] U. N. D. of Economic und P. D. Social Affairs, „World Urbanization Prospects 2018,” New York, 2018. Online verfügbar: <https://population.un.org/wup/Publications/Files/WUP2018-Report.pdf>
- [2] „Linz Innovation,” <https://innovation.linz.at/>, 31.03.2023.
- [3] „ICT4D.at,” <https://www.ict4d.at/>, 31.03.2023.
- [4] „Planet Linz Days,” <https://www.linztourismus.at/freizeit/linz-entdecken/linz-inspiriert/planetlinz/days/>, 31.03.2023.
- [5] „Jugend hackt,” <https://jugendhackt.org/>, 31.03.2023.
- [6] „Nochba - Jugend hackt Österreich 2022,” YouTube video, 2017, 31.03.2023. Online verfügbar: <https://www.youtube.com/watch?v=85AwJrCyK78>
- [7] MPreneur, „MPreneur International Training,” <https://mpreneur.myouth.eu/international-training/>, 31.03.2023.
- [8] United Nations, „Sustainable Development Goals,” <https://www.un.org/sustainabledevelopment/sustainable-development-goals/>, 31.03.2023.
- [9] „mPreneur Website - Nochba,” <https://mpreneur.myouth.eu/nochba/>.
- [10] J. B. L. Julianne Holt-Lunstad, Timothy B. Smith, „Loneliness and Social Isolation,” <https://journals.sagepub.com/doi/abs/10.1177/1745691614568352>.
- [11] „neighbourhood characteristics and loneliness among older adults,” <https://jech.bmj.com/content/75/5/464>.
- [12] S. Johnson, J. Bacsu, T. McIntosh, B. Jeffery, und N. Novik, „COVID-19 and Seniors Food Access, Health, and Well-being: A Rapid Review,” Canadian Journal of Dietetic Practice and Research, 2021. Online verfügbar: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8127016/>
- [13] „Nochba - Gemeinschaft leben, teilen und nachhaltig gestalten,” <https://www.immotopia.at/immotopia-2023/nochba/>, 31.03.2023.
- [14] ICT4D.at, „ICT4D.at Project-Forge 2 Report,” <https://www.ict4d.at/2022/11/ict4d-at-project-forge-2-report/>, 2022, [Zugriff: 31. März 2023].
- [15] United Nations Educational, Scientific and Cultural Organization, „UNESCO Internet4Trust Conference Programme,” <https://www.unesco.org/en/internet-conference/programme>, 2023, 31.03.2023.
- [16] „Spusu Innovation Award,” <https://award.spusu.at/>, 31.03.2023.
- [17] W. S. Awards, „World Summit Awards,” 2023. Online verfügbar: <https://wsa-global.org/>
- [18] Statista. Worldwide software developer working hours.

- [19] Google LLC. (2021) Flutter. Online verfügbar: <https://flutter.dev/>
- [20] „React Native,” Version 0.68. Online verfügbar: <https://reactnative.dev/>
- [21] „Apache Cordova,” Version 11.0.0. Online verfügbar: <https://cordova.apache.org/>
- [22] „Ionic,” Version 6.0.10. Online verfügbar: <https://ionicframework.com/>
- [23] „Xamarin,” Version 5.10. Online verfügbar: <https://dotnet.microsoft.com/apps/xamarin>
- [24] 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>
- [25] Flutter Team. (2021) Architectural overview. Online verfügbar: <https://docs.flutter.dev/resources/architectural-overview>
- [26] Dart Team. (2021) Flutter packages on pub.dev. Online verfügbar: <https://pub.dev/>
- [27] Google. (2021) Firebase. Online verfügbar: <https://firebase.google.com/>
- [28] Google LLC. (2021) Cloud Firestore. Online verfügbar: <https://firebase.google.com/docs/firestore/client/libraries>
- [29] ——. (2021) Firebase Authentication. Online verfügbar: <https://firebase.google.com/docs/auth>
- [30] Firebase. Security Rules | Firebase. Online verfügbar: <https://firebase.google.com/docs/rules>
- [31] M. Hausleitner. Firestore Security Rules Nochba. Online verfügbar: <https://github.com/Martin-Hausleitner/Nochba/blob/main/firestore.rules>
- [32] ——. Storage Security Rules Nochba. Online verfügbar: <https://github.com/Martin-Hausleitner/Nochba/blob/main/storage.rules>
- [33] Google LLC. (2021) Cloud Firestore. Online verfügbar: <https://firebase.google.com/docs/firestore>
- [34] ——. (2021) Data model. Online verfügbar: <https://firebase.google.com/docs/firestore/data-model>
- [35] ——. (2021) Cloud Storage. Online verfügbar: <https://firebase.google.com/docs/storage>
- [36] Google. Firebase Cloud Functions. Online verfügbar: <https://firebase.google.com/docs/functions>
- [37] „iOS App Store,” <https://www.apple.com/ios/app-store/>.
- [38] „Xcode,” <https://developer.apple.com/xcode/>.
- [39] „Apple Developer Program,” <https://developer.apple.com/programs/>.
- [40] „Signulous,” <https://www.signulous.com/>.
- [41] J. H. Felix Krause, „fastlane,” <https://fastlane.tools/>, 31.03.2023.
- [42] F. Team. Flutter Interactive. Online verfügbar: <https://docs.flutter.dev/development/ui/interactive>

- [43] 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>
- [44] Diego Velásquez. (2021) get. Online verfügbar: <https://pub.dev/packages/get>
- [45] „Mobile Scanner,” https://pub.dev/packages/mobile_scanner.
- [46] Hampton, „Kernnetzwerke, soziale Isolation und neue Medien: Internet- und Mobiltelefonnutzung, Netzwerkgröße und -vielfalt,” 2011. Online verfügbar: https://www.researchgate.net/profile/Keith-Hampton/publication/234164594_Core_Networks_Social_Isolation_and_New_Media_Internet_and_Mobile_Phone_Use_Network_Size_and_Diversity/links/0912f50fad7a0d2f0b000000/Core-Networks-Social-Isolation-and-New-Media-Internet-and-Mobile-Phone-Use-Network-Size-and-Diversity.pdf
- [47] C. Szyperski, „Component-Oriented Programming,” 1998. Online verfügbar: https://www.researchgate.net/publication/227304972_Component-Oriented_Programming
- [48] Google, „ML Kit,” <https://developers.google.com/ml-kit>.
- [49] „Feedback,” <https://pub.dev/packages/feedback>.
- [50] V. Choudhary, „Shake: Flutter Plugin for Detecting Shake Events,” <https://pub.dev/packages/shake>.
- [51] „Twitter,” <https://twitter.com/>.
- [52] „Instagram,” <https://www.instagram.com/>.
- [53] „TikTok,” <https://www.tiktok.com/>.
- [54] „Nebenan,” <https://nebenan.de/>.
- [55] „Dribbble,” <https://dribbble.com/>.
- [56] „Mobbin,” <https://mobbin.design/>.
- [57] „Syncfusion_flutter_datepicker,” https://pub.dev/packages/syncfusion_flutter_datepicker.
- [58] „Remixicon,” <https://github.com/Remix-Design/remixicon>.
- [59] „Flutter-Package,” <https://pub.dev/packages/remixicon>.
- [60] „Material Icons,” <https://fonts.google.com/icons?icon.set=Material+Icons>.
- [61] Rasmus Andersson, „Inter Schriftart,” <https://rsms.me/inter/>.
- [62] „nochba.at,” <https://nochba.at>.
- [63] „nochba.com,” <https://nochba.com>.
- [64] „Mein Bezirk,” https://www.meinbezirk.at/linz/c-wirtschaft/linzer-nachwuchs-hacker-beweisen-sich-in-nordmazedonien_a5654095.
- [65] „Linz,” https://www.linz.at/medienservice/2022/202203_114704.php.
- [66] „OÖNachrichten,” <https://www.nachrichten.at/oberoesterreich/linz/von-linz-nach-nordmazedonien;art66,3728292>.

- [67] „DorfTV,” <https://jugendhackt.org/video/nochba/>.
- [68] „Github Repo,” <https://github.com/Martin-Hausleitner/Nochba>.
- [69] „Instagram,” <https://www.instagram.com/nochba.at/>.
- [70] „Framer,” <https://framer.com>.
- [71] „GoDaddy.” Online verfügbar: <https://www.godaddy.com/de-at>
- [72] „Calculate distance, bearing and more between two Latitude/Longitude points,” <https://www.movable-type.co.uk/scripts/latlong.html>.
- [73] Firebase. (2021) GeoFirestore. Online verfügbar: <https://geofirestore.com>
- [74] Firebase Team. (2021) Firebase Core. Online verfügbar: https://pub.dev/packages/firebase_core
- [75] ——. (2021) Firebase Authentication. Online verfügbar: https://pub.dev/packages/firebase_auth
- [76] ——. (2021) Cloud Firestore. Online verfügbar: https://pub.dev/packages/cloud_firestore
- [77] ——. (2021) Firebase Storage. Online verfügbar: https://pub.dev/packages/firebase_storage
- [78] ——. (2021) Cloud Functions. Online verfügbar: https://pub.dev/packages/cloud_functions
- [79] Sankalan Karunaratne. (2021) OpenAI Client. Online verfügbar: https://pub.dev/packages/openai_client
- [80] Algolia. (2021) Algolia. Online verfügbar: <https://pub.dev/packages/algolia>
- [81] Flutter Map Team. (2021) Flutter Map. Online verfügbar: https://pub.dev/packages/flutter_map
- [82] Google LLC. (2021) Google ML Kit Translation. Online verfügbar: https://pub.dev/packages/google_mlkit_translation
- [83] ——. (2021) Google ML Kit Language Identification. Online verfügbar: https://pub.dev/packages/google_mlkit_language_id
- [84] Flutter Team. (2021) Google Sign In. Online verfügbar: https://pub.dev/packages/google_sign_in
- [85] Pavan Kumar. (2021) Feedback. Online verfügbar: <https://pub.dev/packages/feedback>
- [86] Iman Khoshabi. (2021) Mobile Scanner. Online verfügbar: https://pub.dev/packages/mobile_scanner
- [87] Flutter Team. (2021) HTTP. Online verfügbar: <https://pub.dev/packages/http>
- [88] Syncfusion Inc. (2021) Syncfusion Flutter Datepicker. Online verfügbar: https://pub.dev/packages/syncfusion_flutter_datepicker
- [89] Flutter Team. (2021) Image. Online verfügbar: <https://pub.dev/packages/image>
- [90] ——. (2021) Image Picker. Online verfügbar: https://pub.dev/packages/image_picker

- [91] Karuna Negi. (2021) Image Editor Plus. Online verfügbar: https://pub.dev/packages/image_editor_plus
- [92] Flutter Team. (2021) Cupertino Icons. Online verfügbar: https://pub.dev/packages/cupertino_icons
- [93] LoneWolfTech. (2021) Get Time Ago. Online verfügbar: https://pub.dev/packages/get_time_ago
- [94] Namecheap E-Mail Hosting. Online verfügbar: <https://www.namecheap.com/>
- [95] M. Hausleitner, „VSCode-Erweiterung "Copyminify" für Dart-Code Minimierung,” <https://github.com/Martin-Hausleitner/vscode-copyminify>.
- [96] ——. (2023) VSCode-Erweiterung zur Anzeige der Anzahl von Tokens in Code-Files mit GPT-3 und Codex. <https://marketplace.visualstudio.com/items?itemName=MartinHausleitner.vscode-tokenizer-gpt3-codex>.

Abbildungsverzeichnis

1	Linz hACkT 2022	4
2	Projektpräsentation für den Bürgermeister von Linz und die Magistratsdirektorin am 30. Januar 2023	5
3	ICT4D.at X mPreneur Contest Austria 2022	6
4	Jugend hackt Österreich 2022	8
5	mPreneur School (Ohrid, Nord Mazedonien) X WSA 2022	9
6	17 Sustainable Development Goals - United Nations (Quelle: [8])	10
7	Immotopia Innovation Award 2023	13
8	ICT4D.at Project-Forge Workshop 2022 (Wien)	15
9	UNESCO Internet4Trust Konferenz 2023 (Paris):	16
10	Spusu Innovation Award 2023 (Wien)	17
11	Screenshot der Nochba-App im Google Play Store	20
12	ClickUp Dashboard Kalender	21
13	Statistiken zur plattformübergreifenden Nutzung von Statista [18]	24
14	Vergleich der mobilen Frameworks nach Stand 3.2022 [19, 20, 21, 22, 23]	25
15	Systemarchitektur Diagramm	29
16	Anmeldeprozess	48
17	Login Ansicht auf der App	49
18	Registrierungsprozess	50
19	Registrierungsprozess in der App	52
20	Ansicht der Kategorien und Unterkategorien auf der App	54
21	Feed-Page und Kommentarsektion auf der App	55
22	Meldeansicht für einen Beitrag auf der App	56
23	Schritt: 2 Beitrag erstellen	56
24	Menüfilteransicht auf der App	59
25	Hauptfilteransicht auf der App	60
26	Filtern der Reichweite	60
27	Testbenutzer für die Entwickler	63
28	Öffentliche Profilansicht auf der App	73
29	Profilbearbeitungsansicht auf der App	74
30	Meldeansicht für ein Profil auf der App	74
31	Benachrichtigungsansicht auf der App	75
32	Einladencode-Seite in der App	76
33	Nochba Prototyp in Framer	81
34	Nochba Prototyp in Adobe XD	82
35	Nochba Design System	83
36	Nochba Farbpalette	87
37	App Icons	87
38	Die Schriftfamilie Inter	88
39	Finales Logo	89
40	Logo Design Ideen	90
41	Abbildung mehere Thumbs Zones in der App	91

42	Screenshots verschiedener Versionen des App-Home-Screen	92
43	Screenshot der Nochba Website: nochba.at	93
44	Das erste Datenmodell	97
45	Das neue Datenmodell	100
46	Reichweiten verschiedener Nachbarn	113
47	Verwaltungslogik Diagramm	119

Quellcodeverzeichnis

1	Security Rules Beispiel	33
2	Beispiel zum Einsatz von Fastlane	41
3	Beispiel zum Einsatz von einem GetxController in Kombination mit GetView	44
4	Beispiel zum Einsatz von einem GetxService	46
5	Beispiel von einem Binding	47
6	OpenAI GPT-3.5-turbo API Call	57
7	parseTags von KI output	57
8	Aufnahmeprozess für ein Foto	68
9	Prozess der Bildauswahl und -verarbeitung	70
10	Anzeige des Profilbildes	71
11	uploadProfileImageToStorage Funktion	105
12	getDistanceFromLatLonInMeters Funktion	110
13	incrementLikeCounterOfPost Funktion	115
14	decrementLikeCounterOfPost Funktion	116
15	incrementLikeCounterOfComment Funktion	116
16	decrementLikeCounterOfComment Funktion	117