NearbyChat

Tim Lock, Linus Kurze

Hochschule Osnabrück  
Fakultät Ingenieurwissenschaften und Informatik  
Barbarastr. 16, D-49076 Osnabrück  
tim.lock@hs-osnabrueck.de  
linus.kurze@hs-osnabrueck.de

19. Januar 2023

**Zusammenfassung**

Die App NearbyChat ermöglicht eine verbindungslose Kommunikation in Form von Chats zwischen mehreren Nutzern in der Nähe an. Nutzer erstellen Profile, und getätigte Chats können gespeichert werden. Für die Realisierung der Kommunikation zwischen den Geräten wird ein Mesh, basierend auf Bluetooth Low Energy Advertisements, aufgebaut. Dadurch wird die geringe Reichweite von Bluetooth deutlich erweitert.

# Einleitung

[TODO]

* Aufbau der Arbeit

# Konzept

Das Konzept besteht aus mehreren Teilen: Das Basiskonzept, welches die grundlegende Idee der App beschreibt, das Design mit Logo und Namen und das das Interaktionskonzept, welches die Interaktion des Nutzers mit der App beschreibt. Darüber hinaus gibt es das Konzept der Bluetooth-Kommunikation zwischen den Geräten und ein Datenmodell zur Abbildung der zu persistierenden Daten.

## Basiskonzept

Das Basiskonzept von „NearbyChat“ sieht vor, dass mehrere Smartphones über Bluetooth Low Energy Advertisements ein Mesh bilden. Über dieses Mesh soll ein Austausch von Nachrichten erfolgen. Diese Nachrichten sollen zur Einordnung den Zeitpunkt des Absendens enthalten. Darüber hinaus sollen auch Profile ausgetauscht werden. Diese Profile enthalten erste Informationen über den Nutzer. Sie werden genutzt, um verfügbare Nutzer oder auch gespeicherte Chats anzuzeigen. Ein Profil beinhaltet einen Namen und eine kurze Beschreibung. Aufgrund der Tatsache, dass sich über Bluetooth Low Energy Advertisements nur wenige Daten in ausreichend schneller Zeit übertragen lassen, ist der Inhalt von Nachrichten auf 64 Zeichen begrenzt. Der Name im Profil ist auf 16 und die Beschreibung auf 32 Zeichen begrenzt. In den meisten Profilen wird ein Profilbild erwartet. Da dies aber aufgrund der geringen Übertragungsrate nur sehr langsam übertragen werden könnte, ist eine Profilfarbe eingeführt worden. Diese bietet bei geringem Übertragungsaufwand eine hohe Individualität.

Neben den oben beschriebenen technischen Anforderungen soll die App ergonomisch sein: Die App soll einfach und intuitiv bedienbar sein. Außerdem soll zwischen einem hellen und einem dunklen Design, sowie den Sprachen Deutsch und Englisch gewählt werden können.

Ein diesem Basiskonzept ähnliches Konzept ist den Autoren zum Zeitpunkt des Verfassens dieser Arbeit nicht bekannt. Es gibt zwar viele Dating-Apps, wie zum Beispiel Tinder, die auch den Standort beschränken können, allerdings handelt es sich dabei um Geodaten und nicht um eine vom Gerät aus lokale Suche nach verfügbaren Nutzern [1]. Darüber hinaus nutzt Tinder zur Übertragung der Daten das Internet mit eigenen Servern und nicht Bluetooth als direkte Kommunikation zwischen den Geräten [1]. In der Funktion etwas ähnlicher sind Apples AirDrop und das Pendant Nearby Share von Android. Sie verwenden Bluetooth und WLAN zur Übertragung von Daten, wie Fotos oder Videos [2] [3]. Sie können theoretisch auch zur Übertragung von Nachrichten genutzt werden, jedoch ist die Übertragung von Daten das eigentliche Ziel.

## Design, Logo & Name

Das Design der App umfasst ein grundlegendes Farbschema und ein Logo. Das Farbschema umfasst ein helles Rot[[1]](#footnote-1) und ein dunkles Rot[[2]](#footnote-2). Neben diesen Primärfarben sind im Design nur Weiß, Schwarz und Grautöne zugelassen. Mit Ausnahme der Profilfarben werden auch keine weiteren Farben verwendet. Rot als Designfarbe hat sich aus mehreren Gründen als vorteilhaft erweisen: Zum einen nutzt kein bekannter großer Messenger die Farbe Rot: Facebook Messenger und Signal sind blau [4] [5], WhatsApp ist grün [6], Threema ist schwarz/grün [7] und Snapchat ist gelb [8]. Zum anderen bietet die Farbe Rot eine Vielzahl passender Assoziationen: Rot ist unter anderem die Farbe der Liebe und Leidenschaft [9]. Dies passt gut in den Kontext des Kennenlernens in beispielsweise einem Club oder auf einer Party.

Die Primärfarben finden sich auch im Logo der App wieder. Es besteht aus zwei Sprechblasen, die in den beiden Primärfarben eingefärbt sind. Die eine Sprechblase ist von links ausgerichtet, die andere von rechts. Dies symbolisiert den Austausch zweier Personen und damit exakt die Kernfunktion der App. Der Hintergrund ist weiß und das Logo weist keine weiteren Elemente oder Verzierungen auf. Das Logo ist daher auch in kleineren Größen leicht zu erkennen und bietet trotzdem ausreichende Alleinstellungsmerkmale gegenüber den Logos oben erwähnten Messengern.



Abbildung 1: Logo der App

Der Name „NearbyChat“ beschreibt direkt die Hauptfunktion der App: Das kommunizieren mit sich in der Nähe befindlichen Menschen. Der Begriff „Chat“ oder „Chatten“ findet sich bereits im deutschen Sprachgebrauch wieder [10] [11] und ist daher trotz der Sprache Englisch auch für Deutschsprachige leicht verständlich. Darüber hinaus bietet der Name einen guten Wiedererkennungswert und könnte aufgrund der Internationalität auch in anderen Ländern eingesetzt werden.

## Interaktionskonzept

Die Interaktion des Nutzers mit der App beginnt mit dem Starten. Unmittelbar danach wird eine Art Splash-Screen angezeigt. Während deren Anzeige werden zunächst sämtliche Berechtigungen eingeholt. Darüber hinaus werden, wenn nicht bereits geschehen, Bluetooth und die Ortungsdienste eingeschaltet, damit die App vollständig funktionsfähig ist. Das Einschalten und der Zugriff auf die Ortungsdienste wird, ganz ähnlich zur Corona-Warn-App, benötigt [10]. Android fordert diese Berechtigungen, da über Bluetooth prinzipiell Informationen über den Standort erhoben werden können [11]. Falls die App feststellt, dass sie nicht mit dem Gerät kompatibel ist, da es kein Bluetooth Low Energy Advertising unterstützt, wird dies ebenfalls auf dem Splash-Screen angezeigt.

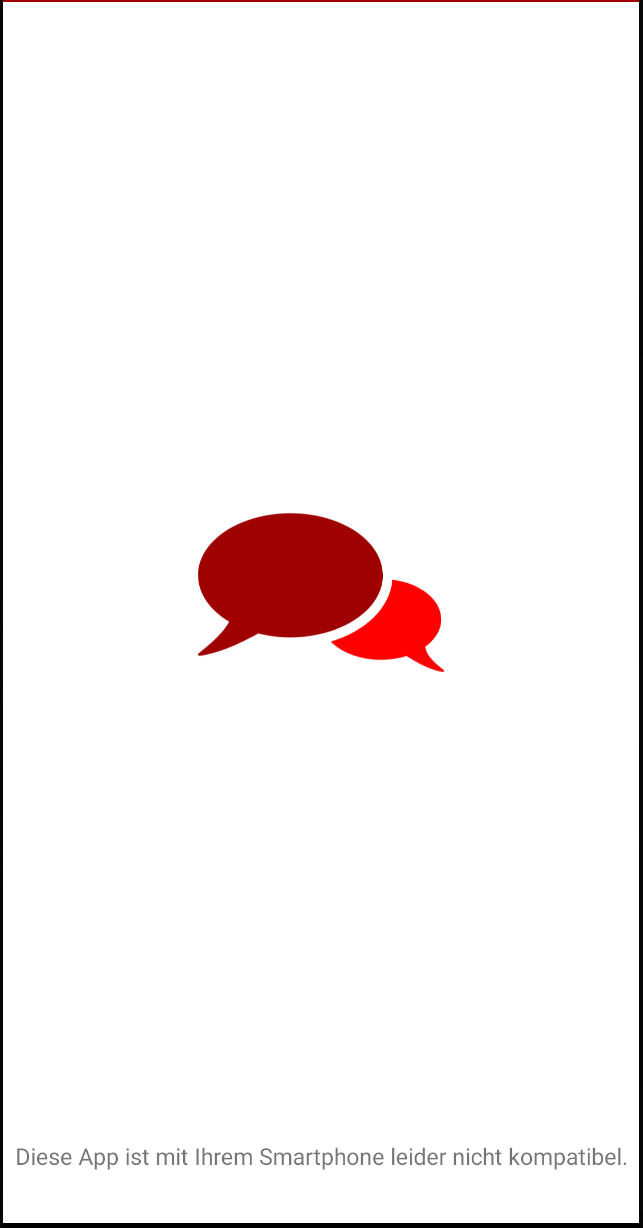


Abbildung 2: Splash-Screen auf inkompatiblem Gerät

Wenn das Gerät mit der App kompatibel ist, werden nach dem Splash-Screen in einer Navigationsleiste am unteren Ende des Bildschirms vier Tabs angezeigt: Der Tab „Verfügbar“, welcher die aktuell verfügbaren Nutzer anzeigt, der Tab „Chats“, welcher die gespeicherten Chats mit anderen Nutzern anzeigt, der Tab Profil, in dem sich das eigene Profil konfigurieren lässt und der Tab „Einstellungen“, in dem sich die Einstellungen der App anpassen lassen. Die Wahl für die Art der Navigation zwischen diesen verschiedenen Ansichten der App ist auf Tabs gefallen, da diese gut geeignet sind, um unabhängige Informationen und Interaktionsmöglichkeiten, inhaltlich in entsprechende Tabs zusammengefasst abzubilden [12, p. 522]. Des Weiteren sind Tabs gut geeignet, wenn die Verwendungsreihenfolge der Inhalte variieren kann, was in diesem Fall ebenso gegeben ist [12, p. 522].



Abbildung 3: Tabs am unteren Bildschirmrand

Der Inhalt der Tabs wird jeweils auf dem Bildschirm zwischen der Navigationsleiste und einer sich am oberen Bildschirmrand befindlichen Informationsleiste angezeigt. Neben der farblichen Hervorhebung des aktuell ausgewählten Tabs ist in der Navigationsleiste unter dem App-Namen eine Kurzbeschreibung des angewählten Tabs zu sehen.

Der Tab „Verfügbar“ zeigt eine sich ständig aktualisierende Profilliste der aktuell erreichbaren Nutzer, jeweils mit Namen, Farbe und Beschreibung. Die Profilfarbe wird mithilfe eines Symbols, einem generischen Avatar auf einer Standortmarkierung angezeigt. Diese sind nach Empfangsstärke sortiert, welche darüber hinaus jeweils durch ein Symbol an der rechten Seite angezeigt wird.



Abbildung 4: Oberer Teil des Tabs "Verfügbar"

Der Tab „Chats“, der die gespeicherten Chats anzeigt, ist hingegen nach dem Zeitpunkt der letzten Interaktion sortiert. Hier wird, neben dem Profilnamen und dem Zeitpunkt der letzten Interaktion, die aktuelle Empfangsstärke des jeweiligen Profils angezeigt. Zur einfachen Unterscheidung von dem Tab „Verfügbar“ ist die Profilfarbe auf einem anderen Symbol, nämlich einer Sprechblase, die mit Text gefüllt ist, angezeigt. Am linken Rand der Profile wird ein roter Punkt angezeigt, wenn eine neue empfangene Nachricht mit diesem Profil ungelesen ist. In diesem Tab können einzelne Chats durch Ziehen von links nach rechts gelöscht werden. Da dies eine kritische Aktion ist, wird sie dem Nutzer durch ein Banner am unteren Bildschirmrand bestätigt. Dieses Banner beinhaltet darüber hinaus eine Schaltfläche, um das Löschen rückgängig zu machen.

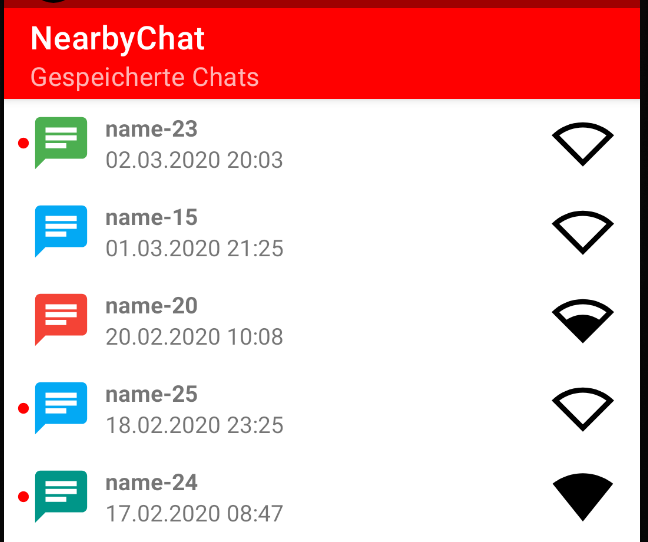


Abbildung 5: Oberer Teil des Tabs "Chats"

Der Tab „Profil“ beinhaltet die Konfiguration des eigenen Profilnamens und der Beschreibung durch Texteingabefelder. Diese sind zur leichten Verständlichkeit mit „Name“ und „Beschreibung“ betitelt. Darunter befindet sich die Konfiguration der Profilfarbe. Ein großes Feld unter der Überschrift Farbe zeigt die aktuell ausgewählte Farbe an. Darunter befinden sich 10 Farbfelder, aus denen sich die Farbe auswählen lässt. In diesem Tab werden vor Änderungen immer die aktuell gespeicherten Profileigenschaften angezeigt. Ein großer Button mit dem Text „Speichern“ ermöglicht das Speichern des eigenen Profils. Nach dem Speichern wird, durch ein Banner am unteren Bildschirmrand, der Vorgang bestätigt.

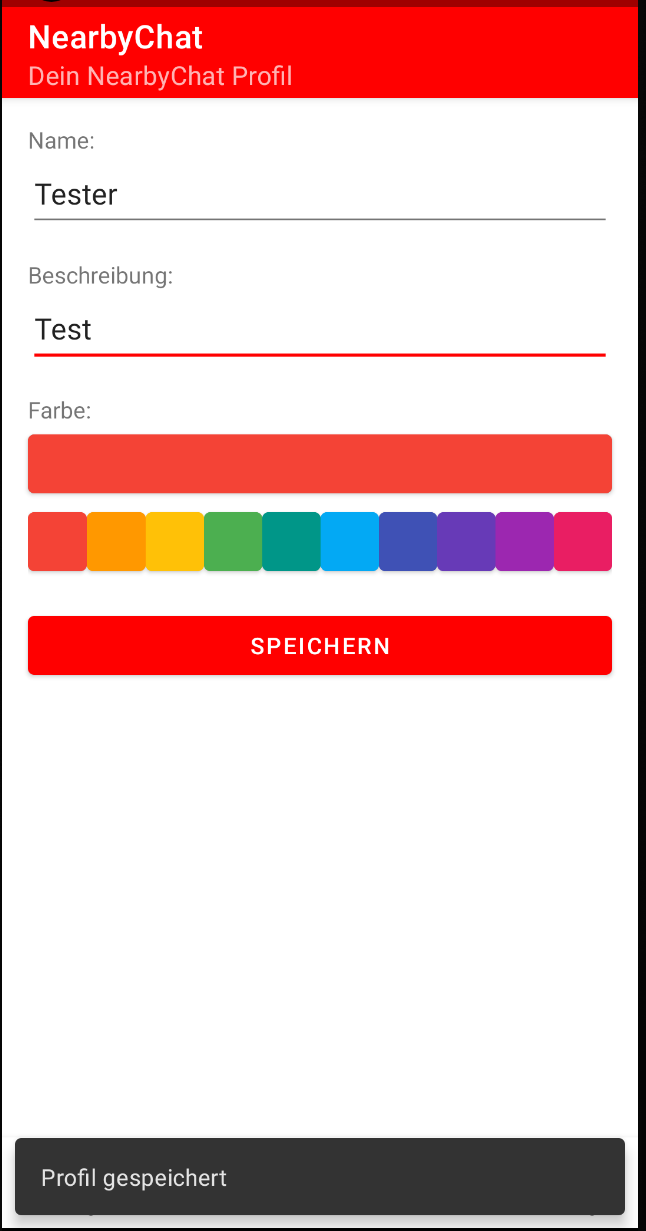


Abbildung 6: Der Tab "Profil" nach Speichern der Eingaben

Der Tab „Einstellungen“ beinhaltet Auswahlmöglichkeiten zur Darstellung und zur Sprache. Unter der Überschrift „Darstellung“ werden die Optionen „Dunkler Modus“, „Heller Modus“ und „Systemstandard“ als Optionsfelder angezeigt. Unter „Sprache“ sind es die Optionsfelder „Englisch“, „Deutsch“ und „Systemstandard“. Es kann für jede Einstellung immer nur ein Optionsfeld angewählt sein. Darüber hinaus wird immer die aktuell angewandte Einstellung angezeigt. Unter der Spracheinstellung folgt noch ein Hinweis, dass die Spracheinstellungen erst nach einem Neustart der App angewendet werden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 7: Konfigurationen im Tab "Einstellungen"

Aus den Tabs „Verfügbar“ und „Chats“ lässt sich durch Antippen eines Profils der Chat mit diesem Profil öffnen. Durch Verwendung des „Zurück“-Buttons von Android gelangt man zurück. In der Informationsleiste am oberen Bildschirmrand befindet sich auch hier, neben dem App-Namen eine Kurzerklärung der aktuellen Ansicht. Direkt darunter befindet sich eine weitere Leiste, die das Profil des Users, deren gemeinsamer Chat geöffnet ist, mit Namen, Farbe, Beschreibung und Empfangsstärke anzeigt. Am unteren Ende des Bildschirms befindet sich ein Eingabefeld, für neue Nachrichten. Direkt rechts neben dem Eingabefeld befindet sich ein Button zum Senden einer Nachricht. Der verbleibende Platz wird genutzt, um den Nachrichtenverlauf anzuzeigen. Die Nachrichten werden von neu, am unteren Ende nach alt, am oberen Ende sortiert. Dabei werden eingehende Nachrichten links, mit der Uhrzeit rechts und ausgehende Nachrichten rechts, mit der Uhrzeit links angezeigt. Die Farbe des Nachrichtenfelds variiert ebenfalls, wenn es sich um eine ausgehende oder eingehende Nachricht handelt. Zwischen den Nachrichten wird, nur wenn es sich geändert hat, das Datum angezeigt. Bei ausgehenden Nachrichten wird, immer nach der Zustellung, unten rechts ein kleiner Haken angezeigt. Standardmäßig sind die Nachrichten nach ganz unten, zu den neusten Nachrichten gescrollt. Wenn der Nutzer weiter nach oben gescrollt hat, wird rechts unten ein Button zum Herunterscrollen angezeigt. Des Weiteren wird neben diesem Button, wenn eine neue Nachricht ankommt und der Nutzer nicht ganz heruntergescrollt hat, ein roter Punkt als Hinweis angezeigt.

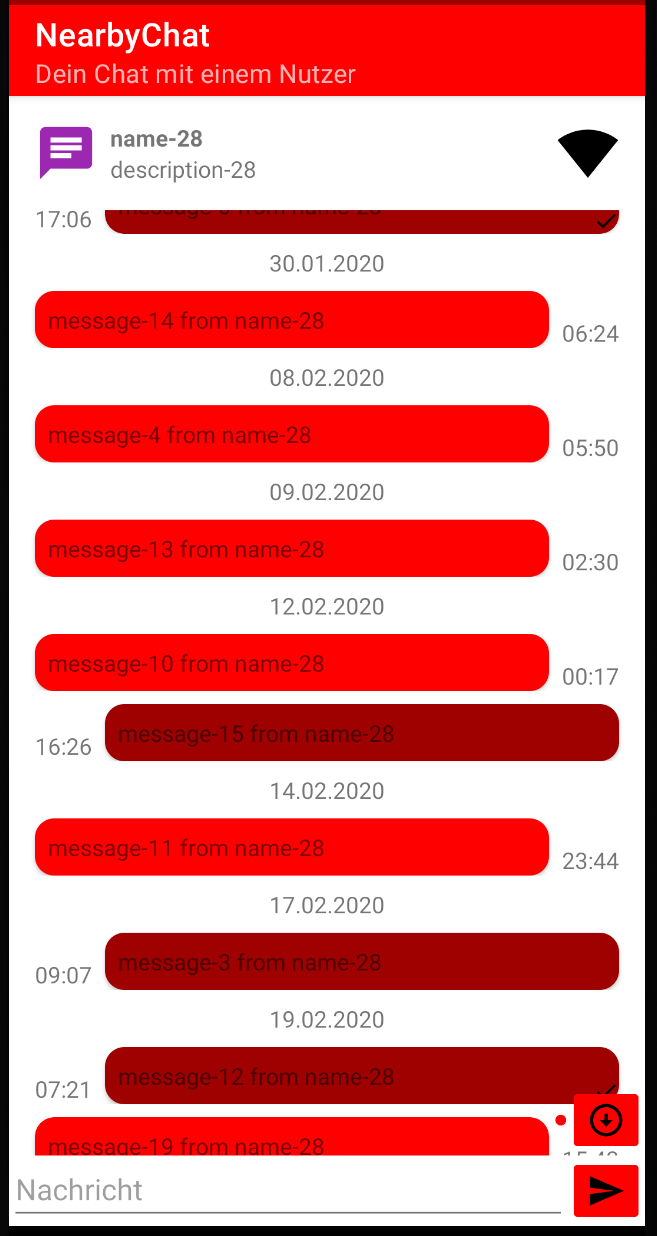


Abbildung 8: Chat mit einem Nutzer

Im Großen und Ganzen orientiert sich das Design stark an etablierten Designs von Messengern, wie WhatsApp [6], Signal [4], Facebook Messenger [5] oder Threema [7]. Das Design ist responsiv und funktioniert daher unter verschiedenen Auflösungen. Auch das Wechseln zwischen Portrait und Landscape ist ohne Verlust eingegebener Daten oder der Scroll-Position möglich.

## Bluetooth Mesh Networking

Die Protokolle zum Austausch von Nachrichten orientieren sich an dem Standard Bluetooth Mesh. Bluetooth Mesh wurde im Sommer 2017 von der Bluetooth Special Interest Group (SIG) eingeführt [15, p. 1]. Die Technologie soll vor allem in den Bereichen Smart Buildings und Industrial IoT Verwendung finden [15, p. 1]. Der Bluetooth Low Energy Stack (BLE) auf welchem Bluetooth Mesh aufbaut, wird von den allermeisten Smartphones und Tablets unterstützt, Bluetooth Mesh jedoch nicht [15, p. 17]. Die verwendete Verbindungstopologie von BLE ist Peer-to-Peer, wobei zwei Geräte eine Verbindung aufbauen, und Nachrichten austauschen. Bei Bluetooth Mesh wird eine Many-to-Many Topologie realisiert, auch genannt Mesh-Topologie, bei welcher jedes Gerät mit jedem anderen Gerät kommunizieren kann.

Aufgrund der fehlenden Unterstützung der für Smartphones, realisiert die App eigene Protokolle für die Bildung eines Meshes und dem Austausch von Nachrichten. Zum Austausch von Nachrichten wird jedoch nicht das Verbindungsorientierten Generic Attribute Profile (GATT) [16, p. 61] verwendet, sondern das Generic Access Profile (GAP) [16, p. 72]. GAP wird vor allem verwendet, um ein Gerät für andere BLE-Geräte sichtbar zu machen, auch Advertising genannt oder diese zu finden Scanning, um anschließend eine Verbindung aufzubauen. Advertising kann jedoch auch verwendet werden, um Datenpakete, auch Advertisementdata genannt, in regelmäßigen Abständen zu senden, welche von jedem Gerät in Reichweite empfangen werden können. Die maximale Größe eines Datenpakets beträgt jedoch 1650 Bytes [17].

Zur Kommunikation werden von der App drei Arten von Nachrichten versendet: Neighbour, Message und Acknowledgement. Nachrichten werden generell immer mit einer geschweiften Klammer begonnen, anschließend folgt Character für den Typ der Nachricht und ein Doppelpunkt. Weitere Nachrichtenfelder werden mit einem Semikolon getrennt. Die Nachricht wird mit einer geschweiften Klammer geschlossen. Um den Nachrichten Durchsatz zu erhöhen, werden Nachrichten in Paketen gebündelt und gemeinsam gesendet. Jedes Bündel von Nachrichten erhält einen Charachter als ID, welcher das Bündel identifiziert. Ein Bündel kann abgeschnittene Nachrichten enthalten, welche vom Empfänger vorgehalten werden und anhand der ID wieder zusammengesetzt werden können.

Damit Geräte eine Nachricht empfangen können benötigen sie eine eindeutige Adresse. Android empfiehlt als Best-Practice IDs zu verwenden, welche vom Benutzer zurückgesetzt werden können [18]. Daher verwendet die App zum Adressieren von Geräten die Android\_ID welche anhand des Gerätes, des Android Nuterkontos und des app-signing-key gebildet wird [19]. Die Android\_ID kann mit einem Factory-Reset des Smartphones zurückgesetzt werden und ist daher geeignet als Adresse.

Die Neighbour-Nachricht orientiert sich an der Heartbeat Message im Bluetooth Mesh. Die Heartbeat Message wird in regelmäßigen Abständen gesendet und signalisiert anderen Geräten, dass das Gerät immer noch erreichbar ist [15, p. 25]. Wie der Abbildung unter dem Absatz zu entnehmen, enthält die Neighbour-Nachricht auch Informationen über das Profil, den Absender der Nachricht und die verbleibenden Hops. Ein Gerät sendet in regelmäßigen Abständen alle ihm bekannten Geräte. Die Neighbour-Nachricht kann sich auf diesem weg theoretisch im ganzen Mesh verbreiten, wird aber durch die maximale Anzahl an Hops begrenzt. Mit jedem weiterleiten einer Neighbour-Nachricht, wird das Feld mit den Hops einmal dekrementiert, erreicht der Wert 0 wird die Nachricht verworfen. So soll verhindert werden, dass Nachrichten an Geräte mit zu großer Entfernung gesendet werden, da die Dauer einer Übertragung mit jedem weitere Hop stark zunimmt. Der Time To Live Ansatz im Bluetooth Mesh verfolgt denselben Ansatz [15, p. 25].



Abbildung 9 Neighbour-Nachricht

Die Message-Nachricht wird verwendet, um Textnachrichten zwischen zwei Geräten auszutauschen. Wie der Abbildung unter dem Absatz zu entnehmen, besitzt eine Nachricht, neben der Adresse von Sender und Empfänger, auch eine Adresse für den nächsten Hop, welcher die Nachricht weiterleiten soll. Im Gegensatz zu Bluetooth-Mesh verwendet das Mesh der App keinen Managed Flooding Ansatz für Message-Nachrichten, bei welchem eine Nachricht an alle erreichbaren Geräte gesendet wird. Dies geschieht, um die Belastung des Mesh möglichst gering zu halten.



Abbildung 10 Message-Nachricht

Nachrichten vom Typ Message werden mit einer Acknowledgment-Nachricht bestätigt. Wie der Abbildung unter dem Absatz zu entnehmen ist, erhält diese wie die Message-Nachricht ein Feld mit der Adresse des nächsten Hops. Der Zeitstempel hat denselben Wert wie die erhaltene Message-Nachricht.



Abbildung 11 Acknowledgement-Nachricht

## Service

Klassen wie Activites, Fragments und ViewModel sind vom Android Lebenszyklus abhängig [21]. Aktionen, die zu einem Zustandswechsel im Lebenszyklus führen, können sein: ein Wechsel zu einer anderen App, das Versetzen des Smartphones in den Ruhezustand oder sogar das Drehen des Smartphones vom Vertikalen in den Horizontalen Modus. Alle eben genannten Aktionen führen dazu, dass nicht persistent gehaltene Daten verloren gehen oder der Mainthread der App sogar beendet wird. Das führt zu zwei Problemen: Die App kann nicht durchgehend Nachrichten erhalten, da zwischen den Zustandsübergängen nicht nach Advertisements gescannt wird. Es können nur Nachrichten empfangen werden, wenn sich die App in einem aktiven Zustand befindet. Durch die eingeschränkte Erreichbarkeit ist das Mesh Netzwerk, welches die Geräte aufspannen instabiler, da teilnehmende Geräte häufige ausfallen. Aufgrund dieser Tatsachen ist eine Trennung der Bluetooth Funktionalitäten vom Rest der Anwendung notwendig. Um eine Trennung von App und Bluetooth zu realisieren wird ein Service verwendet [21].

Ein Service ist unabhängig vom Lebenszyklus der App und besteht sogar weiter, wenn die App von Android beendet wurde. Die Langlebigkeit des Service ermöglicht es, permanent Nachrichten zu empfangen und zu senden. Der Service wird als Foreground Service realisiert [22]. Ein Foreground Service ist aktive, bis er vom Benutzer beendet wird. Ein Foreground Service darf nur unter zwei Bedienungen von einer App verwendet werden: Dem Nutzer muss eine Notification angezeigt werden, solage der Service aktiv ist. Im Manifest muss eine Permission für Foreground Services gesetzt werden. Durch die Verwendung eines solchen Service sind Geräte im Mesh-Netzwerk länger erreichbar und das Mesh gewinnt an Stabilität.

## Datenmodell

Für die App sollen Nachrichten und das eigene Profil, sowie fremde Profile persistiert werden. Die Entitäten sind daher „Message“, „OwnProfile“ und „Profile“.

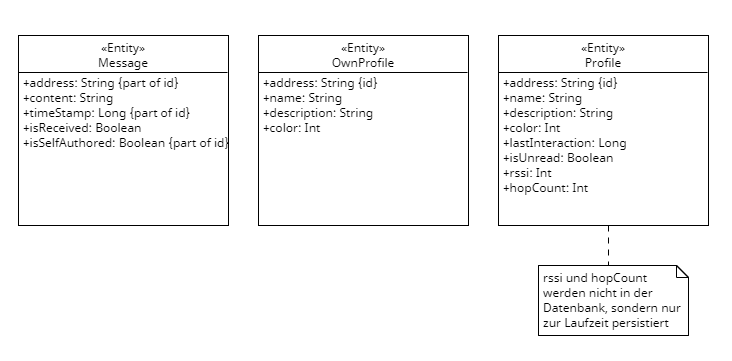


Abbildung 12: Klassendiagramm des Datenmodells

Eine Nachricht enthält neben dem Inhalt einen Zeitstempel und eine Adresse. Die Adresse ist immer die des fremden Profils, an das oder von dem aus Nachrichten gesendet werden. Darüber hinaus wird als Wahrheitswert gespeichert, ob die Nachricht angekommen ist und ob sie selbst verfasst oder vom Gegenüber geschrieben wurde. Der Primärschlüssel setzt sich aus der Adresse, dem Zeitstempel und ob die Nachricht selbstgeschrieben ist, zusammen. Somit können sämtliche Nachrichten eindeutig identifiziert werden, da kein Nutzer mehrere Nachrichten zum exakt selben Zeitpunkt versenden kann. Nachrichten werden vollständig in der Datenbank persistiert.

Das eigene Profil enthält die eigene Adresse, den Namen, die Beschreibung und die Farbe. Die zehn wählbaren Farben werden durch eine Zahl zwischen null und neun einschließlich repräsentiert. Der Primärschlüssel ist die eigene Adresse, da diese immer eindeutig ist. Darüber hinaus wird allerdings nie mehr als ein eigenes Profil gespeichert. Genauso, wie Nachrichten wird das eigene Profil vollständig in der Datenbank gespeichert.

Fremde Profile enthalten, genauso wie das eigene, die Adresse, den Namen die Beschreibung und die Farbe. Darüber hinaus enthalten sie den Zeitstempel der letzten Interaktion, also einer gesendeten oder empfangenen Nachricht und einen Wert, ob es noch ungelesene Nachrichten gibt oder nicht. Ein fremdes Profil umfasst allerdings noch zwei weitere Attribute, die nicht in der Datenbank persistiert werden: Der Hop-Count, also die Anzahl der Geräte auf dem Weg durch das Mesh zum Ziel, und die Received Signal Strength Indication, kurz RSSI, zur Beschreibung der Empfangsstärke. Diese Informationen werden nur zur Laufzeit persistent gehalten.

# Realisierung

Zur Realisierung werden viele Funktionen von Android Jetpack verwendet. Android Jetpack ist eine Sammlung von Libraries für die Entwicklung von Android-Apps. Es bietet viele Funktionen zum einfachen und schnellen Umsetzen einer App, die über Versionsgrenzen von Android hinaus kompatibel ist. Darüber hinaus lässt sich mit Jetpack der Code schlanker halten und Best Practices besser umsetzen [13].

Im Folgenden wird auf die Besonderheiten der Realisierung im Bereich Softwaredesign, User Interface, Service, Bluetooth-Kommunikation und Datenhaltung eingegangen.

## Softwaredesign

Wie sich dem Paket Diagramm unter diesem Absatz entnehmen lässt, ist die Anwendung in drei Pakete aufgeteilt: App, Common und Service.

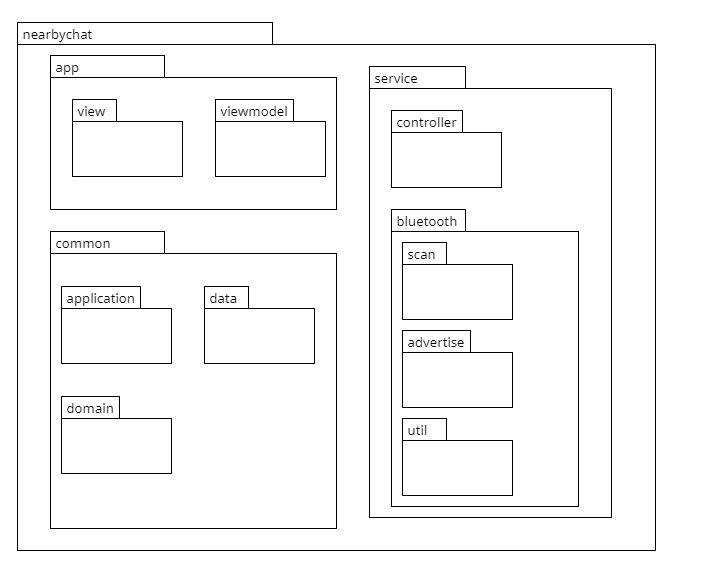


Abbildung 13 Paket Diagramm

Das App-Paket enthält Klassen, die für die Realisierung des User Interface relevant sind. Das Service-Packet hält neben der Service Klasse noch weitere Pakete, welche die Kommunikation mit Bluetooth ermöglichen. Das Common-Paket hält Klassen, welche für die Datenpersistenz relevant sind und sowohl vom App-Paket als auch vom Service-Paket verwendet werden.

[Auf Android Architektur Design Empfehlungen eingehen]

## Realisierung des User Interfaces

Die Besonderheiten bei der Realisierung des User Interfaces beginnen bereits beim Splash-Screen. Dieser wird durch eine dezidierte Aktivität umgesetzt. Er muss die Besonderheiten von Android zur Anforderung von Permissions und zur Initiierung von Änderungen an den Einstellungen beachten: Wenn die App Permissions über das System anfordern lässt, werden diese nur einmalig durch das System abgefragt und bei Ablehnung durch den Benutzer auch bei erneuter Initiierung nicht erneut abgefragt. Daher muss die App nach dem Versuch der Anforderung von Permissions über das System eine alternative Behandlung für fehlende Permissions aufweisen. Diese wird konkret durch eine Popup-Benachrichtigung mit Verlinkung zu den entsprechenden Einstellungen umgesetzt. Die Aktivierung von Bluetooth erfolgt durch eine Anfrage an das System, welches nach Bestätigung des Nutzers Bluetooth aktiviert. Für die Ortungsdienste hingegen bietet das System keine Möglichkeit der Abfrage in dieser Form. Daher muss auch hier eine eigene Popup-Benachrichtigung erstellt werden, die auf die entsprechende Stelle in den Einstellungen verweist. Permissions und Änderungen von Einstellungen werden immer nur dann gefordert, wenn sie nicht bereits erteilt bzw. angepasst sind. Sie werden allerdings auch so lange gefordert, bis sie erteilt bzw. angepasst sind, da die App ohne sie nicht funktionsfähig ist. Die Popup-Benachrichtigungen werden durch den „AlertDialog“ von Android Jetpack realisiert.

Die Ansicht mit den vier Tabs „Verfügbar“, „Chats“, „Profil“ und „Einstellungen“ wird in einer einzigen Aktivität umgesetzt. Hierbei ist hervorzuheben, dass die Navigation durch die „BottomNavigationView“ von Googles Material Design realisiert wird. Material Design bietet, unter Anderen für Android, viele Komponenten, Guidelines und Best Practices im Bereich User Interface Design [14]. Die Änderung des Inhalts und somit der eigentlichen Tabs wird durch Fragmente aus Android Jetpack umgesetzt. Sie bieten die Möglichkeit andere Inhalte beziehungsweise Layouts zu laden, ohne eine neue Aktivität aufrufen zu müssen. Des Weiteren wird die „ActionBar“ von Android Jetpack zur Realisierung der Informationsleiste am oberen Bildschirmrand verwendet.

Die Tabs „Verfügbar“ und „Chats“ verwenden in ihren Fragmenten sogenannte „RecyclerViews“ zur Darstellung der Listen von Profilen. „RecyclerViews“ sind ebenfalls ein Teil von Android Jetpack. Für die beiden Tabs ist jeweils ein Adapter implementiert, der auf Änderungen an den Daten reagiert, indem er diese visualisiert. Des Weiteren bietet der Adapter einen Click-Listener an. Dadurch kann beim Klicken auf ein Profil der Chat mit dem entsprechenden User geöffnet werden.

Das Fragment des Tabs „Chats“ implementiert einen Listener, der auf Zieh-Gesten reagiert. Dieser wird durch den „ItemTouchHelper“ von Android Jetpack zur Verfügung gestellt. Durch diesen Listener kann das Entfernen von Chats realisiert werden. Ein Banner, realisiert durch die „SnackBar“ des Material Design, bestätigt das Löschen. Die „SnackBar“ ermöglicht darüber hinaus durch Antippen eines Textes das Rückgängig-Machen des Löschens.

Die Fragmente zu den Tabs „Profil“ und „Einstellungen“ verwenden die androideigenen „ScrollViews“, damit die Inhalte auch bei verschiedenen Displayhöhen und im Portrait- sowie Landscape-Modus angezeigt werden können.

Im Fragment des Tabs „Profil“ wird der „SavedInstanceState“ verwendet, um die Profilfarbe bei erneutem Laden des Fragments anzeigen zu können. Dies geschieht beispielsweise bei dem Wechsel von Portrait- zu Landscape-Modus oder andersherum. Andere Änderungen des Nutzers, wie zum Beispiel Eingaben von Texten in die Textfelder werden automatisch beibehalten. Die Zeichenbegrenzung des Namens und der Beschreibung ist durch eine entsprechende Eigenschaft der Textfelder umgesetzt. Das Speichern des Profils wird mit einer „SnackBar“ aus dem Material Design bestätigt.

Die Einstellungen, die im gleichnamigen Tab getätigt werden können, werden durch das Fragment an die darunter befindliche Aktivität weitergeleitet. Diese nutzt die androideigenen „SharedPreferences“, um die Einstellungen jeweils als Key-Value-Paar zu speichern und wendet diese auch an. Die Anwendung geschieht bei den Sprachen durch „Locale“ und bei dem Farbschema durch „AppCompatDelegate“.

Die Chats mit einem User werden in einer eigenen Aktivität geöffnet. Diese verwendet ein „ConstraintLayout“ von Android Jetpack, um sowohl die zweite Leiste am oberen Bildschirmrand als auch das Texteingabefeld, den Senden-Button und den Herunterscroll-Button korrekt positionieren zu können. Auch hier ist die Zeichenanzahl der Nachricht durch eine entsprechend gesetzte Eigenschaft des Texteingabefelds beschränkt. Das Profil des anderen Users wird durch die Aktivität bei Änderung aktualisiert. Die Chatnachrichten werden durch einen „RecyclerView“ realisiert, wie in den Tabs „Verfügbar“ und „Chats“. Der dazugehörige Adapter übernimmt neben Visualisieren von Änderungen, wenn Nachrichten dazu kommen, auch die Aufgabe, die Nachricht auf der richtigen Seite, mit der richtigen Farbe anzuzeigen, je nachdem, ob sie ein- oder ausgehend ist. Außerdem wird durch Prüfen des Zeitstempels der vorangegangenen Nachricht das Datum nur bei einer Änderung angezeigt. Wenn das Datum nicht angezeigt werden muss, kollabiert das Textfeld für das Datum, sodass es keinen Platz benötigt. Die Aktivität kann durch einen Scroll-Listener beim Scrollen die aktuelle Scroll-Position abfragen, und dadurch prüfen, ob der Button zum Herunterscrollen und bei Bedarf auch der Punkt als Indikator für eine neue Nachricht angezeigt werden sollen. Darüber hinaus behält die Aktivität die Scroll-Position immer bei, es sei denn es wird eine neue Nachricht empfangen, und die Scroll-Position war vorher ganz unten. In diesem Fall scrollt die Aktivität erneut nach ganz unten, sodass die neue Nachricht zu sehen ist. Darüber hinaus hebt die Aktivität den Ungelesen-Vermerk des Profils auf, wenn die neusten Nachrichten gesehen wurden.

## Realisierung des Service

Ein Service wird realisiert, indem die Service Klasse von Android von einer eigenen Klasse erweitert wird. Zur Kommunikation mit dem Service wird eine eigene Klasse verwendet, welche einen Service startet, beendet oder Methoden im Service aufrufen kann. Mit einem Service kann auf zwei Arten kommuniziert werden, entweder Nachrichtenbasiert mit Intents oder mit einem direkten Zugriff auf die Instanz, welcher mit dem Binden an den Service ermöglicht wird. Die Kommunikation wird mit einem Binding realisiert, welches ermöglicht Methoden direkt aufzurufen. Initial wird ein Service immer als Background Service erstellt, durch den Aufruf der Service Methode startForeground() wird der Service zu einem Foreground Service. Da die App vom Service benachrichtigt werden muss, wenn neue Geräte erreichbar sind, wird diese Richtung der Kommunikation mit Intents realisiert. Die Klasse, welche die Kommunikation mit dem Service steuert, implementiert einen Broadcastreceiver, um entsprechende Intents vom Service zu erhalten.

## Realisierung der Bluetooth-Kommunikation

Aus dem zweiten Kapitel ergeben sich zwei Anforderungen an die Bluetooth Kommunikation: Die Kommunikation muss verbindungslos sein. Die Verbindung muss eine Many-to-Many Kommunikation ermöglichen. Daher wird auf die Verwendung von Entitäten des GATT, welche in der Regel zur Kommunikation mit Bluetooth Low Energy verwendet, werden verzichtet. Alternativ zu GATT werden die beiden Klassen BluetoothLeAdvertiser und BluetoothLeScanner welche Entitäten des GAP auf Android Klassen abbilden verwendet.

Der BluetoothLeAdvertiser lässt sich mit mehreren Werten parametrisieren. Unter anderem kann der Sendungsintervall verändert werden oder die Stärke der Übertragung, wodurch die Übertragungsrate und Reichweite beeinflusst werden. Damit der BluetoothLeAdvertiser neben Informationen zum Smartphone auch Daten der App übertragen kann, müssen diese als AdvertiseData gesetzt werden. AdvertiseData implementieren das Interface Parcelable welches von Android zum Serialisieren verwendet wird. Zum Speichern von Daten benötigt AdvertiseData einen Key, welcher im Falle der App eine 128 Bit UUID ist. Der BluetoothLeAdvertiser kann nur eine bestimmte Größe an Daten versenden. Die konkrete Größe wird beim Initialisieren berechnet.

Der BluetoothLeScanner lässt sich ebenfalls mit einigen Werten parametrisieren. Neben dem Scanintervall können Ergebnisse des Scans auch gefiltert werden. Gefundene Advertisements werden über ein Callback erhalten. Mithilfe des Scanfilters können Geräte ausgeschlossen werden, welche nicht über die vom BluetoothLeAdvertiser verwendete UUID verfügen.

Die Verwaltung sämtlicher Bluetooth Funktionalitäten erfolgt über die Klasse MeshController, welcher die einzige Schnittstelle für den Service ist. Der MeshController steuert welche Daten gesendet werden und wie auf eingehende Nachrichte reagiert werden muss. Eingehende Neighbour-Nachrichten werden in einer eigenen Klasse zum Speichern von Profilinformationen gehalten. Message-Nachrichten werden vorgehalten, falls diese unvollständig sind, weitergeleitet oder mit dem Absenden einer Acknowledge-Nachricht bestätigt. Nachrichten welche unbestätigt sind werden ebenfalls in einer designeierten Klasse vorgehalten. Bekannte Nachbarn müssen regelmäßig auf Inaktivität, welche nach 5 Sekunden eintrifft, geprüft werden. Unbestätigte Nachrichten müssen ebenfalls in regelmäßigen Abständen erneute gesendet werden, bis zur Bestätigung. Um dies zu erreichen, wird ein ScheduledExecutor verwendet. Ein ScheduledExecutor ist ein Thread, welche in Regelmäßigen Abständen eine Aktion ausführen kann. Im falle der App werden alle 5 Sekunden inaktive Nachbarn verworfen und unbestätigte Nachrichten erneute gesendet.

Zum regelmäßigen Versenden von Nachrichten wird die Klasse AdvertisementExecutor verwendet, welche Ebenfalls einen ScheduledExecutor nutzt, allerdings mit einem niedrigeren Intervall als im MeshController. Der AdvertisementExecutor baut in jedem Intervall ein Paket von Nachrichten welches abgesendet wird sobald die maximal Größe erreicht wurde oder keine Nachrichten mehr zur Verfügung stehen. Es werden abwechselnd Neighbour-Nachrichten oder Message- und Acknowledgement-Nachrichten versendet, so wird verhindert, dass bei zu vielen Neighbour-Nachrichten keine Message-Nachrichten mehr gesendet werden können.

## Realisierung der Datenhaltung

Die Realisierung der Datenhaltung arbeitet auf zwei Ebenen: Zum einen werden die Daten, die in der Datenbank persistiert werden sollen mit der Library „Room“ aus Android Jetpack in eine lokale Datenbank geschrieben. Zum anderen werden die zur Laufzeit zu persistierenden Daten mit „LiveData“, ebenfalls ein Teil von Android Jetpack, persistent gehalten. „LiveData“ ermöglicht es, einfach auf Änderungen zu reagieren. Da „LiveData“ auch mit Room kompatibel ist, werden auch die Daten aus der Datenbank in der App in „LiveData“ verwaltet. Änderungen an diesen Daten müssen allerdings durch gesonderte Funktionen auf der Datenbank ausgeführt werden und können nicht an den „LiveData“ Objekten vorgenommen werden. „LiveData“ bietet für die Daten der Datenbank darüber hinaus den Vorteil, dass die Zugriffe auf die Datenbank lose gekoppelt sind, sodass die Blockade von Android, dass nicht vom UI-Thread synchron auf die Datenbank zugegriffen werden kann, nicht greift.

Besonders ist das Repository bei der Umsetzung der Datenhaltung hervorzuheben. Es verwaltet mit Hilfe sogenannter Data Access Objekt, kurz DAOs, die Daten aus der Datenbank und führt Änderungen asynchron über die DAOs auf der Datenbank aus. Darüber hinaus verwaltet es aber auch die zur Laufzeit persistenten Daten und aktualisiert sämtliche Daten, wenn nötig: Kommt beispielsweise eine Nachricht an, oder wird gesendet, so wird geprüft, ob das zugehörige Profil bereits gespeichert ist. Wenn dies nicht der Fall ist, wird es aus den verfügbaren Profilen übernommen und gespeichert, oder wenn auch dies nicht möglich ist, wird ein leere Platzhalterprofil erstellt. Ist die Nachricht eine eingehende Nachricht, so wird die Markierung für ungelesene Nachrichten auf dem Profil gesetzt. Sämtliche gespeicherten Profile werden durch das Repository aktualisiert, wenn sie verfügbar sind und die verfügbaren Profile unterschiede gegenüber den gespeicherten aufweisen. Dies betrifft nur die Eigenschaften Name, Beschreibung, Farbe, RSSI und Anzahl der Hops, nicht jedoch den letzten Interaktionszeitpunkt oder die Markierung für ungelesene Nachrichten. Ändert sich ein gespeichertes Profil, so wird dieses, wenn verfügbar, immer um die RSSI und die Anzahl der Hops ergänzt, sodass auch in gespeicherten Chats die aktuelle Empfangsstärke angezeigt werden kann.

Das ViewModel erfüllt ebenfalls hervorhebenswerte Aufgaben: Zum einen kapselt es die Logik des Repositorys durch einfache Methoden für die verschiedenen Views. Zum anderen ist es für die Verwaltung des Chat-Services verantwortlich. Es stellt sicher, dass der Service gestartet wird und reagiert auf Änderungen durch den Service, beispielsweise wenn ein neues Profil entdeckt wird oder eines verschwindet. Diese Änderungen reicht das ViewModel an die Datenbank weiter.

# Tests

Zum testen der verschiedenen Komponenten der Anwendung wurden zwei Arten von Tests durchgeführt. Zum einen wurde das User Interface inklusive der Datenhalten mehreren Probanden vorgeführt. Zum anderen wurden die Bluetooth Funktionalitäten mithilfe von Instrumented Tests welche auf einem Emulator ausgeführt werden und dem JUnit Framework.

## Tests des User Interfaces und der Datenhaltung

Die Tests des User Interfaces wurden mit Tests der Datenhaltung kombiniert: Es wurden Methoden, zum Laden von Testdaten implementiert. Diese Daten wurden durch das UI dargestellt und konnten auch bearbeitet, ergänzt oder gelöscht werden. Durch diese Testdaten war ein Testen bereits vor Fertigstellung der Bluetooth-Kommunikation möglich.

Die konkreten Tests wurden dann mit mehreren Personen an verschiedenen Smartphones mit aufgespielter App und den Testdaten umgesetzt. Dabei haben die Personen sich zunächst allein zurechtfinden und ein paar Aufgaben erfüllen sollen. Beispiele für solche Aufgaben sind: Das eigene Profil anpassen, eine Nachricht an einen Nutzer versenden, oder einen Chat mit einem Nutzer löschen. Beim Erfüllen dieser Aufgaben haben die Personen alles erwähnt, dass ihnen aufgefallen ist. Dadurch konnte unter Anderem identifiziert werden, dass die ursprünglich verwendete Markierung von ungelesenen Chats durch einen roten Hintergrund, als unschön und schlechter lesbar empfunden wurde, weshalb der kleine rote Punkt am Rand eingeführt wurde. Darüber hinaus fiel der Bedarf nach einem Button zum Herunterscrollen in den Chats auf, und dass das eigene Profil ohne das Ermöglichen des Scrollens im Landscape-Modus nicht auf jedem Gerät vollständig bearbeitet werden kann, da der Speichern-Button abgeschnitten war.

Auch nach Fertigstellung der Bluetooth-Kommunikation wurden Tests durch Anwendung durchgeführt. Hierbei haben zwei Personen über die App kommuniziert oder ihr Profil angepasst und es wurde geprüft, ob die Änderungen oder neuen Nachrichten, wie erwartet dargestellt werden. Auch die Markierung für ungelesene Chats wurde getestet. Bei diesen Tests zeigte sich nur kleiner Korrekturbedarf an der Datenhaltungslogik.

## Tests der Bluetooth-Kommunikation

[TODO]

# Installation

[TODO]

* Beachten: min API, Kompatibilität (meldet die App aber auch)

# Zusammenfassung & Ausblick

[TODO]

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | Tinder, „Tinder,“ 12 Januar 2023. [Online]. Available: https://play.google.com/store/apps/details?id=com.tinder&hl=de&gl=US. [Zugriff am 16 Januar 2023]. |
| [2] | N. Hery-Moßmann, „So funktioniert AirDrop - einfach erklärt,“ Chip, 23 Juli 2022. [Online]. Available: https://praxistipps.chip.de/so-funktioniert-airdrop-einfach-erklaert\_99963. [Zugriff am 16 Januar 2023]. |
| [3] | I. Bauer, „Nearby Share auf Android - so funktioniert's,“ Heise, 16 Mai 2022. [Online]. Available: https://www.heise.de/tipps-tricks/Nearby-Share-auf-Android-so-funktioniert-s-7095779.html. [Zugriff am 16 Januar 2023]. |
| [4] | Signal Foundation, „Signal - Sicherer Messenger,“ 11 Januar 2023. [Online]. Available: https://play.google.com/store/apps/details?id=org.thoughtcrime.securesms&gl=DE. [Zugriff am 16 Januar 2023]. |
| [5] | Meta Platforms Inc., „Messenger,“ 13 Januar 2023. [Online]. Available: https://play.google.com/store/apps/details?id=com.facebook.orca&gl=DE. [Zugriff am 16 Januar 2023]. |
| [6] | WhatsApp LLC, „WhatsApp Messenger,“ 5 Dezember 2022. [Online]. Available: https://play.google.com/store/apps/details?id=com.whatsapp&gl=DE. [Zugriff am 16 Januar 2023]. |
| [7] | Threema GmbH, „Threema,“ 5 Januar 2023. [Online]. Available: https://play.google.com/store/apps/details?id=ch.threema.app&gl=DE. [Zugriff am 16 Januar 2023]. |
| [8] | Snap Inc, „Snapchat,“ 12 Januar 2023. [Online]. Available: https://play.google.com/store/apps/details?id=com.snapchat.android&gl=DE. [Zugriff am 16 Januar 2023]. |
| [9] | A. Soboth, „Farbe Rot: Bedeutung und Wirkung der Farbe Rot,“ Focus Online, 5 Februar 2019. [Online]. Available: https://praxistipps.focus.de/farbe-rot-bedeutung-und-wirkung-der-farbe-rot\_107802. [Zugriff am 16 Januar 2023]. |
| [10] | Duden, „chatten,“ [Online]. Available: https://www.duden.de/rechtschreibung/chatten. [Zugriff am 17 Januar 2023]. |
| [11] | Duden, „Chat,“ [Online]. Available: https://www.duden.de/rechtschreibung/Chat. [Zugriff am Januar 17 2023]. |
| [12] | Stiftung Warentest, „Corona-App - Stand­ort-Zugriff auf Android-Handys,“ 17 Juni 2020. [Online]. Available: https://www.test.de/Corona-App-Standort-Zugriff-auf-Android-Handys-5624423-0/. [Zugriff am 16 Januar 2023]. |
| [13] | Android Developers, „Bluetooth Permissions,“ Google Inc., 12 Januar 2023. [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth/permissions. [Zugriff am 16 Januar 2023]. |
| [14] | W. O. Galitz, The Essential Guide to User Interface Design, Hoboken: Wiley, 2007. |
| [15] | M. Woolley, „Bluetooth,“ 2 Dezember 2020. [Online]. Available: https://www.bluetooth.com/bluetooth-resources/bluetooth-mesh-networking-an-introduction-for-developers/. [Zugriff am 18 Januar 2023]. |
| [16] | M. Woolley, „Bluetooth,“ 6 Juny 2022. [Online]. Available: https://www.bluetooth.com/bluetooth-resources/the-bluetooth-low-energy-primer/?utm\_source=internal&utm\_medium=blog&utm\_campaign=technical&utm\_content=the-bluetooth-low-energy-primer. [Zugriff am 18 Januar 2023]. |
| [17] | Android Developrs, „Android Developrs,“ Google Inc., [Online]. Available: https://source.android.com/docs/core/connect/bluetooth/ble\_advertising. [Zugriff am 2023 Januar 18]. |
| [18] | Android Developers, „Best practices for unique identifiers,“ Google Inc., 12 Januar 2023. [Online]. Available: https://developer.android.com/training/articles/user-data-ids#instance-ids-guids. [Zugriff am 18 Januar 2023]. |
| [19] | Google Developers, „Settings.Secure,“ Google Developers, 8 Juni 2022. [Online]. Available: https://developer.android.com/reference/android/provider/Settings.Secure#ANDROID\_ID. [Zugriff am 18 Januar 2023]. |
| [20] | Android Developers, „The activity lifecycle,“ Google Inc., 07 September 2022. [Online]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle. [Zugriff am 19 Januar 2023]. |
| [21] | Android Developers, „Services overview,“ Google Inc., 9 Januar 2023. [Online]. Available: https://developer.android.com/guide/components/services. [Zugriff am 19 Januar 2023]. |
| [22] | Android Developers, „Foreground services,“ Google Inc., 12 Januar 2023. [Online]. Available: https://developer.android.com/guide/components/foreground-services. [Zugriff am 19 Januar 2023]. |
| [23] | Android Developers, „Android Jetpack,“ Google Inc., [Online]. Available: https://developer.android.com/jetpack. [Zugriff am 16 Januar 2023]. |
| [24] | Material Design, „Material Design,“ Google, [Online]. Available: https://m2.material.io/. [Zugriff am 16 Januar 2023]. |

# Arbeitsaufteilung

|  |  |
| --- | --- |
| *Beitrag* | *Verantwortliche(r)* |
|  |  |
|  |  |
|  |  |
|  |  |

# Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Verwendung anderer als den angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

*oder*

Hiermit erklären wir, dass wir die vorliegende Arbeit selbstständig und ohne Verwendung anderer als den angegebenen Hilfsmittel angefertigt haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Tim Lock)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Linus Kurze)

Osnabrück, den 19. Januar 2023

1. Hexadezimale Darstellung: #FF4B4B [↑](#footnote-ref-1)
2. Hexadezimale Darstellung: #FF9E9E [↑](#footnote-ref-2)