St.-Anna-Schule Wuppertal
Q2 2021
Herr Gerd Heischkamp
Abgabe: 24.04.2021

Programmierung einer React Native App, mit Firebase als Backend und dem Schwerpunkt Datensicherheit

Finn Malkus
finn@malkus.de
IF1 / HEIS
Q2 20/21

Gliederung

1. Vorwort

Dieses Dokument dient als Dokumentation der App "GERD", welche ich im Rahmen meiner besonderen Lernleistung für mein Abitur 2021 programmieren durfte. Das Projekt ist OpenSource und daher auf GitHub zu finden unter: https://github.com/FinnMal/Gerd. Zum Stand der Abgabe ist die App noch nicht im AppStore/PlayStore veröffentlicht, dies ist aber in Planung.

2. Idee

Die Kommunikation innerhalb und außerhalb von Vereinen und Organisationen verläuft meist digital per Mail und in einigen Fällen sogar noch analog per Post oder Flyer. Für einen Verein ist es wichtig in Konakt zu den Mitgliedern zu bleiben, sei es um aktuelle Veranstaltungen zu vermarkten oder die Noten für das nächste Chortreffen per Mail zu verschicken. Der Anwendungsbereich ist vielseitig. Die Kommunikation läuft jedoch ebenso unterschiedlich. Der eine Verein verschickt wichtige Infos per Newsletter, der andere lädt die Notentexte auf ausländischen Servern hoch, die entweder stundenlang beim Download brauchen, oder am nächsten Tag schon nicht mehr erreichbar sind. Und im schlimmsten Fall kommen die Mitteilungen beim falschen Empfänger an. Der Datenschutz bleibt dabei auch der Strecke. Diese Zenarien erleben viele Vereine und Organisationen täglich. Die Prozesse sind lästig für die Vereinsbetreiber, aber auch für die Vereinsmitglieder. Sich durch verschiedne Mails und Websites durchzuwühlen um eine einzige Datei zu finden macht für Vereinsmitglieder keinen Spaß. Zudem gibt es auch keine wirklichen Alternativen, die Kommunikation per WhatsApp fällt zum Beispiel aus Datenschutzgründen weg und andere Messenger-Apps sind für ältere Mitglieder meist zu kompliziert. Die Idee der GERD-App ist es daher ein einheitliches Tool zu bieten, welches all diese Anwendungsbereiche vereint und auf die Kommunikation für Vereine spezialisiert ist. Die App bietet die Möglichkeit per Push-Notifications Mitglieder zu informieren, Termine zu planen, Dateien zu verbreiten und im privaten verschüsseltem Chat mit einzelnen Mitgliedern zu kommunizieren. Das alles auf Grundlage einer einfachen Bedienbarkeit und Anonymität.

3. Strukturierung

Die App besteht aus der React Native Anwendung, welche der Benutzer auf seinem Endgerät installiert und ausführt, dem Firebase Backend, welches auf Google Servern in Iowa, USA läuft, und dem OneSignal Backend, was die Übermittlung der Push-Notifications übernimmt.

Firebase ist ein, in der Regel kostenloses, Tool von Google. Es basiert auf der Funktionalität der Google Cloud Server und bietet ein einfaches Backend für Web- und Mobileanwendungen. Für das Projekt wurde die Firebase-Echtzeitdatenbank sowie die sogenannten "Firebase Functions" verwendet.

3.1 Firebase Datenbank

Die "Firebase Realtime Databse" basiert auf einer JSON-Struktur und kann sowohl per REST und direkt über die JavaScript Erweiterung von Firebase abgefragt werden. Die Requests auf die Datenbank setzten immer eine vorherige Authentifizierung vorraus (Siehe 8.2, Datensicherheit: Firebase). Mit Firebase ist die Regrestirerung per E-Mail, Facebook, Google oder anonym möglich. Für dieses Projekt wurde die Anoyme Authentifizierung verwendet, um, vorallem für ältere Benutzer, keine zusätlichen Hürden während der Regrestirerung zu verursachen (Siehe 8.1, Datensicherheit: Anonymes Verwenden der App). Der großteil der Kommunikation zu den Firebase-Datenbank-Servern läuft über den "DatabaseConnector" (Siehe 9.1, Besondere Stellen im Code: DatabaseConnector.js)

In der Firebase Datenbank werden Vereinsinformationen, Benutzerdaten, Informationen zu Chats und die verfügbaren Farbschemen gespeichert. Der Zugriff auf die verschiedenen Datensätze ist durch die "Firebase-Database-Rules" beschränkt (Siehe. 8.2., Datensicherheit: Firebase)

Durch sogenannte "Database Listener" kann innerhalb der App in "Echtzeit" auf Änderungen in der Datenbank reagiert werden. Diese Funktion kommt zum Beispiel beim Ändern von Benutzerinformationen, Dateimetadaten oder Veranstaltungsinformationen zum Einsatz. Dies gewähleistet die Korrektheit und aktualität der Daten und Informationen. Nicht selten kommt es jedoch von kleinen Verzögerungen von ca. 1-2 Sekunden, welche jedoch zu vernachlässigen sind. (Siehe 9.1, Besondere Stellen im Code:

DatabaseConnector.js).

3.2 Firebase Functions

Die "Firebase Functions" sind ein programmierbares Backend basierend auf Node.js, welche auf den Google Servern ausgeführt werden. "Firebase Functions" sind mit sogenannten Triggern ausgestattet, welche die erstellen Alogorothmen ausführen. "Trigger" können das Hinzufügen, Entfernen oder Ändern von Einträgen in der Firebase Datenbank sein, bestimmte Intervalle und Uhrzeiten, oder gwöhnliche HTTP-Requests.

Dieser Trigger reagiert zum Beispiel auf neue Mitteilungen eines Clubs:

```
exports.sendMessageNotifications =
functions.database.ref('clubs/{club_id}/messages/{mes_id}').onCreate(async (snapshot, context)
=> {console.log('message added') });
```

Es wird eine neue Funktion mit dem Namen "sendMessageNofitications", welche auf Änderugen innerhalb eines Club-Objekts reagiert. Teile eines Pfades lassen sich durch Variablen ersetzten. In diesem Beispiel wurde die Variable 'club_id' verwendet, damit der Trigger auf jedes Club-Objekt reagiert. Auf die Pfad-Variabeln kann in der Funktionsausführung zugegriffen werden. Bei den Ereignishandlern kann zwischen "onWrite" (Daten werden erstellt, aktualisiert oder gelöscht), "onCreate" (Daten werden erstellt), 'onUpdate' (Daten werden aktualisiert) und "onDelete" (Daten werden gelöscht) ausgewählt werden.

Für dieses Projekt werden die "Firebase Functions" verwendet, um QR-Codes für Vereins-Invites zu generieren (Siehe 6.3., Abläufe: Beitreten eines Vereins), Push-Notifications für zum Beispiel Chat-Nachrichten oder Veranstaltungen zu versenden und sich wiederholende Veranstaltungen zu planen.

3.3 OneSignal

Für das Versenden von Push-Mitteilungen wird der Dienst von OneSignal verwendet. OneSignal ist ein Anbieter zum versenden von Push-Notifications auf Verschiedene Plattformen. Für dieses Projekt wurde die von OneSignal bereitgestellte Erweiterung für React Native Apps verwendet. OneSignal kann mithilfe von HTTP-Requests durch die Firebase Functions angesprochen

werden. Der Empfang der Push-Notifications erfolgt außerhalb der eigentlichen React Native App und wird im Hintergrund ausgeführt, unabhängig davon, ob die App geöffnet ist oder nicht. Klickt der Benutzer auf eine Benachrichtigung, oder wird eine Benachrichtigung empfangen während die App geöffnet ist, kann darauf mithilfe von Callbacks durch die React Native Instanz reagiert werden.

Beispiel für das Erstellen eines OneSignal-Callbacks in App.js:

```
OneSignal.addEventListener("opened", this.onOpened);
```

Jedem Nutzer ist eine eigene OneSignal-ID zugewiesen, welche beim ersten Start der App von OneSignal generiert wird und anschließend in der Firebase Datenbank gespeichert wird (Siehe Abläufe: Anmeldung). Somit lassen sich Push-Notifications an einzelne Nutzer senden.

Die Benutzer können zudem durch Tags identifiziert werden, welche gesetzt werden, sobald ein Benutzer einem Verein beitritt. Durch die Tags werden Vereins-Mitteilungen an die Teilnehmer versendet.

Beispiel für das setzen eines OneSignal-Tags:

```
OneSignal.sendTag("NAME", "VALUE");
```

Beim Senden einer Push-Notification durch die Firebase Functions können diese Tags dann angegeben werden.

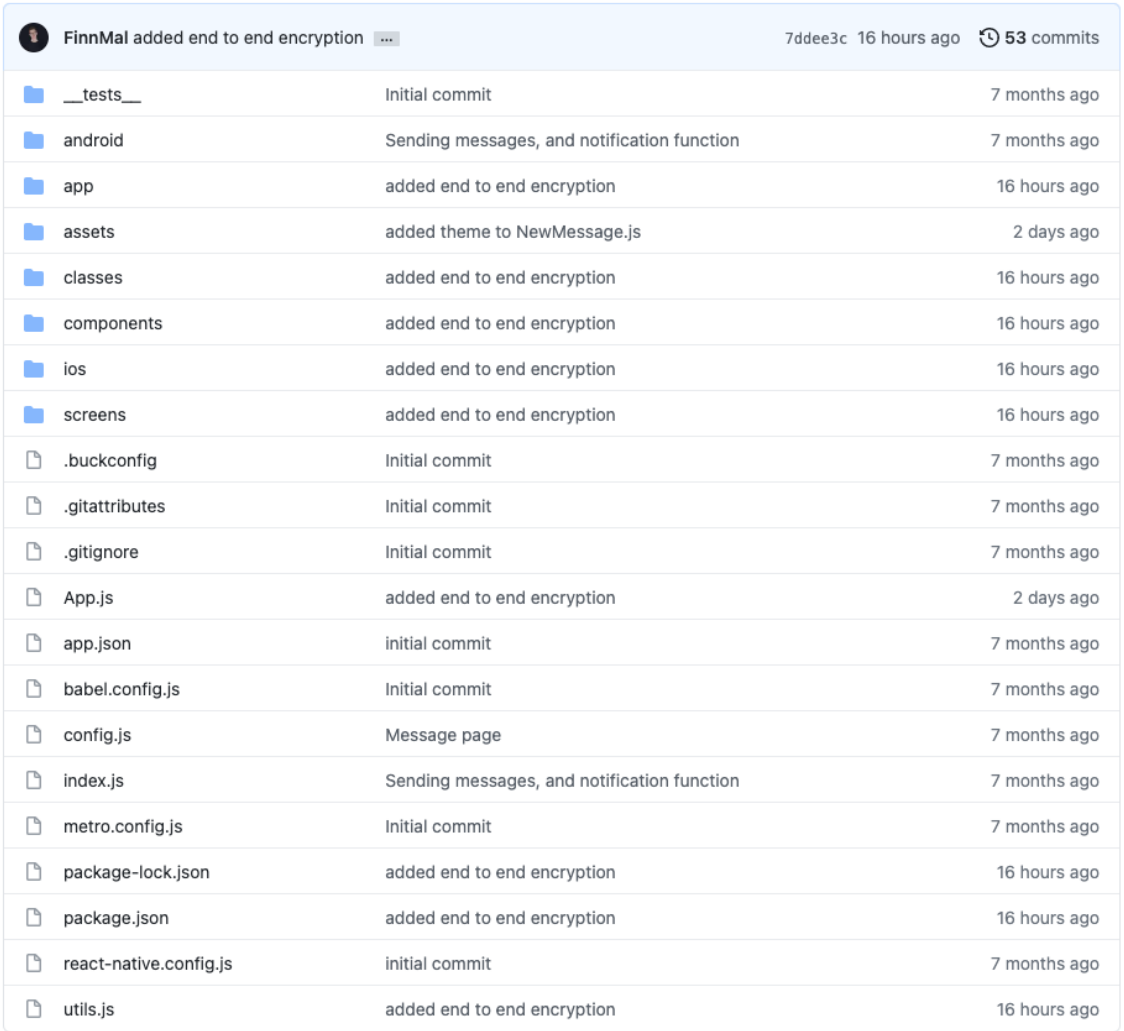
4. React Native

React Native ist ein Framework zum entwickeln von mobilen Anwendungen, gleichermaßen für IOS als auch für Android. Der Quellcode wird ausschließlich in JavaScript geschreiben, und ist für beide Plattformen gleich. Das Framework basiert auf React, der JavaScript-Bibilothek von Facebook zum Erstellen von Benutzeroberflächen. React Native rendert, basierend auf dem JavaScript-Quellcode, native Objekte. Dadurch ist für den Benutzer kein Unterschied zu gewöhnlichen Apps sichtbar. Das Framework wird unteranderem für die Apps von Facebook, Instagram, Skype, Tesla und Pinterrest verwendet. Für die jeweilige Plattform werden von React Native Projekt-Ordner erstellt. Für IOS ein Xcode Projekt und für Android ein Android Studio Projekt. Die Projekte beinhalten den Code zum ausführen des React Native Frameworks, sowie alle

nötigen JavaScript-Packages (Siehe 4.2.3, Entwicklung in der Praxis: Installation eines Pakets). Die erstellten Projekte beinhalten jedoch nicht den JavaScript Quellcode selbst, dieser wird beim ausführen der App aus den js-Dateien gelesen und ausgeführt. Der JavaScript-Code wird von React Native also nicht in Nativen Code umgewandelt. Es lassen sich somit auch Änderungen am Nativen Code vornehmen und APK bzw. IPA Dateien erstellen.

4.1 Struktur des Projektordners

Der Ordner eines React Native Projekts enthält die Quellcodes als JavaScript-Dateien (.js), Konfigurationsdateien (.json), und Ordner für die jeweiligen Plattformen, hier IOS ("ios") und Android ("android"). Die Ordner "app", "classes", "components" und "screens" wurden nicht von React Native erstellt, hier befinden sich weitere JavaScript-Dateien.



FinnMal added end to end encryption ...	7ddee3c 16 hours ago	**53** commits
__tests__	Initial commit	7 months ago
android	Sending messages, and notification function	7 months ago
app	added end to end encryption	16 hours ago
assets	added theme to NewMessage.js	2 days ago
classes	added end to end encryption	16 hours ago
components	added end to end encryption	16 hours ago
ios	added end to end encryption	16 hours ago
screens	added end to end encryption	16 hours ago
.buckconfig	Initial commit	7 months ago
.gitattributes	Initial commit	7 months ago
.gitignore	Initial commit	7 months ago
App.js	added end to end encryption	2 days ago
app.json	initial commit	7 months ago
babel.config.js	Initial commit	7 months ago
config.js	Message page	7 months ago
index.js	Sending messages, and notification function	7 months ago
metro.config.js	Initial commit	7 months ago
package-lock.json	added end to end encryption	16 hours ago
package.json	added end to end encryption	16 hours ago
react-native.config.js	initial commit	7 months ago
utils.js	added end to end encryption	16 hours ago

Die Plattform-Ordner werden einmalig zu Beginn der Entwicklung mit folgendem Befehl erstellt:

```
react-native eject
```

React Native erstellt nun die Projekte für die jeweiligen Plafformen.

4.2 Entwicklung mit React Native

4.2.1 Programmierung

Die Programmierung der eigentlichen App wird in den jeweiligen JavaScript-Dateien vorgenommen. Die App startet in "App.js". Hier wird der "Navigator" bestimmt, also die Art und Weise, wie die verschiedenen Screens miteinander verküpft sind und wie der Benutzer von einem Screen zum anderen navigieren kann. Im Navigator wird auch der Start-Screen bestimmt, also der Screen, der beim öffnen der App als erstes angezeigt wird. Bei diesem Projekt ist der Start-Screen "ScreenHandler.js" (Siehe 9.3., Besondere Stellen im Code: ScreenHandler.js). In "App.js" können zudem weitere Aufgaben erledigt werden, die zur intialisierung der App wichtig sind. In diesem Projekt wird in "App.js" zum Beispiel die Verbindung zu Firebase und OneSignal hergestellt. In der Programmierung gilt es als nächstes die ersten Elemente zu rendern. Hierzu ist es wichitg, die Struktur eines Elements in React Native zu kennen. Diese beginnt zunächst mit dem Konstruktor, in dem die "props" an das Objekt übergeben werden.

Der standart Konstruktor eines React-Elements sieht folgendermaßen aus:

```
constructor(props) {
        super(props);
 }
```

Werden keine Anpassungen im Konstruktor vorgenommen, so kann dieser auch weggelassen werden.

Nach dem erstellen des Elements wird von React die "render"-Methode aufgerufen. Hier wird, ähnlich wie bei HTML, die Strukstur der Elemente angegeben.

Ausschnitt der "render"-Methode von ClubCard.js:

```
render() {
  return (
   <Theme.TouchableOpacity
    style={{
     marginBottom: 20,
     borderRadius: 14,
     padding: 11,
     backgroundColor: "#" + this.props.club_color,
     shadowColor: "#" + this.props.club_color,
     flexDirection: 'row',
     alignItems: 'center',
     justifyContent: 'center'
    }}
    onPress={() => this.onPress()}>
    ...
   </Theme.TouchableOpacity>
  );
 }
```

Deutlich wird, dass es sich bei der "render"-Methode um eine Mischung aus HTML, CSS und JavaScript handelt. Die Objekt-Attribute, wie zum Beispiel "style" oder "onPress" werden dem Objekt durch das "props"-Objekt mitgegeben. Möchte man beispielsweise auf die übergebene "onPress"-Funktion zugreifen, so benötigt man folgenden Code-Ausschnitt:

```
this.props.onPress()
```

Will man den programmierten Code testen, so ist dies zum einen über die Plattform Expo möglich, aber auch über den sogenannten "Metro Bundler" von Facebook. Expo bietet eine Vorauswahl an Packages an und kann innerhalb

der Expo App selbst getestet werden, die eigene App muss somit nicht über Xcode oder Android Studio compiliert werden. Jedoch ist es auch nicht möglich zusätliche Packages zu installieren, die zum Beispiel für die Integration von OneSignal nötig sind. Daher wurde für dieses Projekt der Metro Bundler verwendet. Der Metro Bundler basiert auf Node.js und startet sich automatisch, sobald man die App per Xcode auf dem iPhone oder einem Emulator startet. Öffnet man nun die App, lädt wird der Quellcode über den Metro Bundler heruntergeladen, wie in der Abbildung sichtbar.

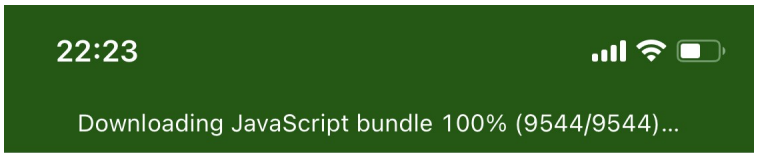


Abbildung 1

Der Metro Bundler reagiert zudem auf änderungen an dem lokalen Projekt und läd automatisch die neue Version auf das Testgerät. Somit muss nicht bei jeder änderung die App neu compiliert und auf dem iPhone installiert werden. Zudem werden Fehlermeldungen direkt in der App angezeigt.

Fehlermeldung der App, nach dem Starten mit dem Metro Bundler:

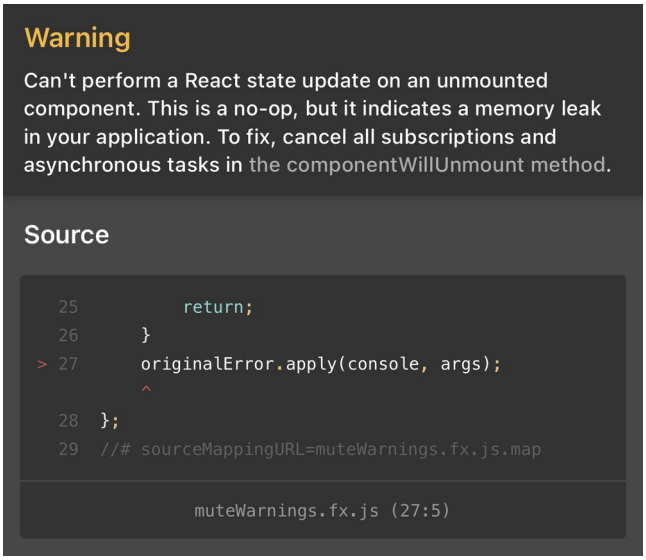


Abbildung 2

Nachdem die App erfolgreich getestet und fehlerfrei ist, kann in Xcode die Release-Version erstellt werden. Diese basiert dann nicht mehr auf dem Metro Bundler und kann als eigenständige App im AppStore angeboten werden.

4.2.2. Installation eines Pakets

Wie auch in anderen Programmiersprachen üblich, kann das eigene Projekt durch die installation externer Pakete erweitert werden. Bei React Native ist hierfür der Node Package Manager (npm) vorgesehen. Pakete können wie üblich über folgenden Befehl installiert werden:

```
npm install (...)
```

Um das Paket nach der installation mit React Native verwenden zu können muss es mit den jeweiligen nativen Projekten verknüpft werden. Beim IOS Projekt ist das mithilfe von "CocoaPods" (pod) möglich. CocoaPods verknüpft die mit npm installierten Pakete mit dem Xcode Projekt. Die Abhängigkeiten, also welche Pakete mit dem Xcode Projekt verknüpft werden müssen, werden in der Datei "Podfile" festgelegt, welche sich im Ordner des IOS-Projekts befindet. In der Regel sind hier aber keine Änpassungen nötig, da die installierten npm-Pakete automatisch ausgelesen und verknüpft werden.

Folgender Befehl startet die Verknüpfung mit CocoaPods, dieser muss innerhalb des IOS-Projektordners ausgeführt werden:

```
pod install
```

Nach der Verknüpfung muss das Xcode-Projekt neu compiliert und auf dem Gerät installiert werden.

4.3. Vor-/Nachteile

Die plattformübergreifende Programmierung ist eines der Vorteile der React Native Plattform. Sie spart zum einen viel arbeit, lässt die App aber auch, durch selbes Design und Funktionen, auf beiden Plattformen einheitlicher wirken. Wird von IOS bzw. Android eine neue Funktion für Apps im Betribsystem veröffentlicht, so dauert es bei React Native jedoch einige Wochen bis Monate, bis es verwendet werden kann. Da man auf ein von der Community erstelltes npm-Paket angewiesen ist. Hier hat man aber auch die Möglichkeit den nativen Code der App selbst zu erweitern und somit die Funktionalität mit seinem React Native Projekt zu verknüpfen. Die Verwendung von React Native setzt Grundlagen vorraus, die die meisten Programmierer bereits mitbringen: HTML, CSS und JavaScript. Laut Satista war JavaScript 2019 auf Platz 3 der

beliebtesten Programmiersprachen. Durch Expo und Metro Bundler ist das Testen und die Fehlerbehebung auf dem eigenen Gerät sogar einfacher als mit den herkömmlichen Tools wie Xcode oder Android Studio. React Native profitiert zudem von einer sehr großen Community, das Projekt wird also stetig weiterentwickelt.

5. Funktionen und Screens in der App

Die App bietet umfangreiche Möglichkeiten zur Komuunikation mit Vereinsmitgliedern. Es ist zwischen den Funktionen für Vereinsbetreiber und Vereinsmitglieder zu unterscheiden. Der Akkount des Vereinsbetreibers ist im Grunde eine erweiterte Version eines Normalen Akkounts. Der Funktionsumfang wird also nur durch die Verwaltungsmöglichkeiten des Vereins erweitert. Spezielle Vereinsakkounts, bei denen die Verwaltung des Vereins im Vordergrund steht, gibt es nicht. Durch das Menü (unten im Bild) kann zwischen den einzelnen Bildschirmen gewechselt werden.

5.1 Startseite

Für alle Benutzer wird auf der Startseite die Übersicht der aktuellen Veranstaltungen und Mitteilungen angeziegt (Siehe Abbildung 3). Oben rechts können die Benutzer auf ihr Profilbild klicken und gelangen somit zu einer Übersicht über die letzten Push-Notifications. Diese Funktion ist jedoch nur in Planung und zum jetzigen Stand noch nicht umgesetzt. Direkt unterhalb des Profilbilds werden die aktuellen Veranstaltungen der Vereine angezeigt. Hierbei handelt es sich um einen sogenannten "Slider". Nach vier Sekunden wird die angezeigte Veranstaltung automatisch nach links rausgeschoben und die nächste Veranstaltung wird angezeigt. Durch wischen kann ebenfalls die nächste Veranstaltung angezeigt werden. Die automatische Bewegung wird damit ausgeschaltet. Als Hintergrund können Vereine eigene Bilder hochladen, oder der Server wählt zufällig eines der Standartmuster aus. Scrollt man nun weiter runter, werden die letzten Mitteilungen der Vereine angezeigt. Hat der Nutzer bereits auf eine Mitteilung geklickt, so wird diese vom Startbildschirm entfernt und kann nach einem Klick auf "Alle anzeigen" in einer extra Übersicht wieder aufgerufen werden.

5.2 Mitteilung

Nach dem Klicken auf eine Mitteilung öffnet sich der Mitteilungs-Screen (Siehe Abbildung 4). Der Benutzer kann nun den gesamten Text der Mitteilung lesen. Scrollt man weiter runter, so lassen sich auf verknüpfte Dateien runterladen, oder Veranstaltungen anzeigen. Oben wird der Author der Mitteilung mit Name und Profilbild angezeigt, sofern dies vom Author selbst so festgelegt wurde. Mitteilungen lassen sich auch anonym verfassen, sodass als Author nur der Verein selbst angezeigt wird. Ist die Anzeige des Authors aktiviert, kann über das Brief-Symbol links vom Namen ein neuer Chat mit dem Verfasser gestartet und eine private Nachricht gesendet werden, sofern man nicht selbst der Verfasser ist.

5.3 Verwaltung

Für Vereinsbetreiber gibt es einen zusätzlichen Bildschirm (Siehe Abbildung 5)

Hier werden alle Vereine angezeigt, bei denen der Benutzer als Administrator eingetragen ist. Druch klicken auf einen bestimmten Verein gelangt man zur Verwaltungsansicht. Oben rechts findet sich der Menüpunkt "Verein beitreten", dieser wird bei Vereinsmitgliedern direkt unten im Menü, anstelle des zweiten Symbols, angezeigt.

Nach der Auswahl eines Vereins gelangt man auf dem Verwaltung-Screen (Siehe Abbildung 6). Hier können Änderungen am Verein vorgenommen werden, wie zum Beispiel Name, Logo oder Farbe. Zudem können Dateien hochgeladen, Events oder Gruppen erstellt und QR-Codes (Siehe 6.3, Abläufe: Beitreten eines Vereins) generiert werden. Die Ansichten der einzelnen Menüpunkte sind ähnlich, daher wird im Folgendem exemplarisch auf die Gruppen-Verwaltung eingegangen.

5.3.1. Gruppen Verwaltung

Durch Gruppen können Vereinsbetreiber Mitteilungen gezielter an verschiedene Mitglieder des Vereins senden (Siehe 6.3, Abläufe: Senden einer Mitteilung). Gruppen können sowohl öffentlich zugänglich als auch privat sein. Während des Beitretens wählt der Benutzer aus den öffentlichen Gruppen aus (Siehe 6.3,

Abläufe: Beitreten eines Vereins). Privaten Gruppen kann nur per Einladungscode und Bestätigung durch den Vereinsbetreiber beigetreten werden. In der Gruppen-Verwaltung können bestehende Gruppen bearbeitet und neue erstellt werden. Zudem wird der Benutzer gewarnt, sobald Sicherheitsrisiken vorliegen (Siehe Abbildung 7).

5.4 Verein beitreten

Auf dem "Beitreten"-Screen können sowohl Vereinsbetreiber als auch Mitglieder einem neuen Verein beitreten (Siehe Abbilung 8). Durch die Suche kann einem öffentlichen Verein beigetreten werden. Verfügt der Nutzer über einen Einlasungscode, so kann er dem Verein beitreten, unabhängig davon, ob dieser öffentlich ist oder nicht. Vereinsbetreiber können Einladungscodes auch in Form von QR-Codes verbreiten, diese lassen sich durch einen Klick auf das Kamera-Sysmbol oben rechts einscannen. Geplant ist auch, dass QR-Codes direkt mit der vorinstallierten Kamerapp eingescannt werden können. Per sogenannten Deeplinks könnte dann direkt zur App weitergeleitet und der zugehörige Verein geöffnet werden. Ist die App nicht installiert, so würde der Benutzer zum entsprechenden Store weitergeleitet und würde nach der Installation und Einrichtung den Screen zum Beitreten sehen. Nachdem ein Verein ausgewählt beziehungsweise ein QR-Code eingescannt wurde, kann der Nutzer Gruppen auswählen, denen er beitreten möchte. Gruppen, die mit dem eingegeben oder eingscannten Einladungscode verknüpft sind, werden automatisch ausgewählt.

5.5. Chat

Auf der Chatübersicht werden alle aktiven Chats angezeigt, jeweils mit dem Namen und dem Profilbild des Chatpartners, der letzten Nachricht, und der vergangenen Zeit seit der letzten Nachricht (Siehe Abbilung 9). Durch klicken auf einen Chat gelangt man auf die Chatansicht (Siehe Abbilung 10). Hier werden die letzten Nachrichten mit Uhrzeit angezeigt. Unterhalb des Chatverlaufs kann man seine Nachricht eintippen und mit einem Klick auf den Button rechts verschicken. Tippt der Chatspartner eine Nachricht ein, so wird dies unterhalb vom Namen im Header angezeigt. Weitere Nachrichten können angezeigt werden, indem man nach oben Scrollt. Um die Internetverbindung

nicht zu stark auszulasten werden ältere Nachrichten aus einer lokalen SQL-Datenbank gelsen (Siehe 9.4., Besondere Stellen im Code: Chat.js)

5.6. Einstellungen

In den Einstellungen kann der Nutzer sein Profilbild bearbeiten, seinen Nutzernamen ändern und seine Vereins-Gruppen bearbeiten oder einen Verein verlassen (Siehe Abbilung 11). Außerdem hat der Benutzer die Möglichkeit seinen Account zu löschen.

6. Abläufe

6.1 Anmeldung

Die Anmeldung in der App erfolgt anonym. Nach einer kurzen Einleitung über die wichtigsten Funktionen der App, kann der Nutzer einen Benutzernamen wählen (Siehe Abbilung 12). Der Benutzername darf aus Sicherheitsgründen nicht länger als 50 Zeichen sein, und keine Sonderzeichen enthalten. Benutzernamen können mehr als ein mal vergeben werden. Der Nutzer wird bereits beim Starten der App in Firebase anonym angemeldet und erhält eine eindeutige Id. Klickt der Nutzer auf den Button "Fertig" (Siehe Abbilung 12, Unten im Bild) wird der Benutzer-Account mit der zuvor zugewiesenen Id in der Firebase Datenbank erstellt. Die Id wird im lokalen Speicher gesichert, um beim nächsten Start wieder drauf zugreifen zu können. Der angegebene Benutzername wird in der Datenbank gespeichert. Die von OneSignal generierte Id wird an die Datenbank übergeben. Zudem werden die öffentlichen und privaten Schlüssel zur Verschlüsselung der Chatnachrichten generiert (Siehe 8.3, Datensicherheit: Ende-Zu-Ende Verschlüsselung). Der öffentliche Schlüssel wird in der Datenbank gespeichert. Der private Schlüssel wird bei Android in "EncryptedSharedPreferences" und bei IOS in "Keychain" gespeichert. Beides sind vom System zur Verfügung gestellte Möglichkeiten um sensible Daten sicher zu speichern.

Nachdem der Benutzer regrestirert ist, wird er zum Startbildschirm weitergeleitet (Siehe 5.1, Funktionen und Screens in der App: Startseite). Da zu diesem Zeitpunkt noch keinen Vereinen begetreten wurde, werden auf der Startseite auch keine Events oder Mitteilungen angezeigt. Statdessen sieht der

Nutzer eine kurze Erklärung (Siehe Abbildung 13).

6.2 Beitreten eines Vereins

Alle Nutzer können einem Verein betreten, hier gibt es keine Unterscheidung zwischen Vereinsbetreibern und Vereinsmitgliedern. Der Beitritt geschiet über den Beitreten-Screen (Abbilung 8). Öffentliche Vereine können per Suche gefunden werden, private Vereine können nur per Einladungscode betreten werden. Einladungscodes bestehen aus einer sechstelligen zufälligen Zahlen und Buchstaben Kombination. Die Codes können manuell in das Suchfeld eingegeben werden, oder als QR-Code eingescannt werden. Die QR-Codes werden in der App von den Vereinsbetreibern zusammen mit den Einladungscodes generiert und können, Beispielsweise im Vereinsgebäude, aufgehangen werden. So können neue Mitglieder unkompliziert beitreten. Die Codes sind zum einen mit dem Verein selbst verknüpft, aber auch mit vorher ausgewählten Gruppen des Vereins. So hat man die Möglichkeit Vereinsmitgliedern, die den Code eingeben oder einscannen, direkt einer (privaten) Gruppe zuzuweisen. Das Einscannen ist auf dem Beitreten-Screen mit dem Kamera-Symbol möglich (Siehe Abbildung 8, oben Rechts). Nach dem Klick auf das Symbol, öffnet sich der Scanner und der QR-Code muss vor die Kamera des Gerätes gehalten werden. Wird ein QR-Code erkannt, so ist der Ablauf gleich wie beim Aufruf über das Suchfeld. Es öffnet sich ein Fenster, in dem der Nutzer die jeweiligen Gruppen auswählt denen er beitreten möchte (Siehe Abbildung 14). Gruppen, die mit dem erkannten Code verknüpft sind, werden automatisch ausgewählt. Sie können vom Nutzer aber auch wieder abgewählt werden. Private Gruppen werden nur angezeigt, sofern sie mit dem Einladungscode verknüpft sind. Mit einem Klick auf "Fertig" werden die beigetretenen Gruppen im Benutzer-Object in der Firebase Datenbank gespeichert. Zudem wird der entsprechende OneSignal-Tag gesetzt, damit die Push-Notifications empfangen werden können (Siehe 3.3, Strukturierung: OneSignal). Möchte der Nutzer einer privaten Gruppe beitreten, so wird dies nicht sofort wirksam. Zunächst wird eine Push-Nofication an die Vereinsbetreiber versendet, welche dann das neue Mitglied bestätigen, oder ablehnen können (Siehe 8.3.2, Datensicherheit: Gruppen-Mitteilungen). Die Mitteilungen und Events des Vereins werden nun auf der Startseite angezeigt.

6.3 Senden einer Mitteilung

Mitteilungen an die Teilnehmer gehören zu den Grundfunktionen der App und können ausschließlich von Vereinsbetreibern versendet werden. Geplant ist auch ein Webinterface über das Mitteilungen versendet und zum Beispiel Dateien hochgeladen werden können. Auch bereits genannte Einstellungen zum Anzeigen des Authors bei Mitteilungen werden im Webinterface verfügbar sein.

Um eine neue Mitteilung zu erstellen klickt man unten in der Navigationsleiste auf das Plus Symbol. Dadurch öffnet sich ein neues Fenster. Hier wird dann der Verein ausgewählt, an welchen die Mitteilung gesendet werden soll (Siehe Abbildung 15). Durch den Button unten rechts im Bild lässt sich auf die nächste Seite navigieren. Auf der darauf folgenden Seite gibt der Vereinsbetreiber die Überschrift, den Anreißer und den Text der Mitteilung ein (Siehe Abbildung 16). Im nächsten Schritt können Veranstaltungen erstellt und Dateien hochgeladen werden, die mit der Mitteilung verknüpft werden. Geplant ist, dass hier auch auf bereits erstellte Inhalte des Vereins zurückgegriffen werden kann. Zudem kann ein Beitragsbild hochgeladen werden. Im letzten Schritt wählt der Betreiber die Gruppen aus, an die die Mitteilung gesendet werden soll. Hier kann aus den bestehenden Gruppen ausgewählt werden. Zusätzlich können Gruppen miteinander verlinkt werden (Siehe Abbildung 14). Bei miteinander verlinkten Gruppen wird die Mitteilung nur an die Mitglieder gesendet, die gleichzeitig in beiden Gruppen angemeldet sind. Anhand eines Beispieles wird diese Funktionsweise deutlicher: Der Vereinsbetreiber hat zwei Gruppen eingerichtet. Eine Gruppe "Freizeit 2021" in der sowohl Mitglieder als auch Mitarbeiter dieser Freizeit angemeldet sind. Zudem gibt es eine Gruppe "Mitarbeiter" in der alle Mitarbeiter des Vereins gelistet sind. Nun soll eine Mitteilung gesendet werden, allerdings nur an alle Mitarbeiter der "Freizeit 2021". Durch das Verknpüpfen beider Gruppen werden genau die gewünschten Mitglieder ereicht, da sie zum empfangen der Mitteilung sowohl in der Gruppe "Freizeit 2021" als auch in der Gruppe "Mitarbeiter" gelistet sein müssen.

Nachdem die Inhalte und Empfänger der Mitteilung bestimmt wurden, klickt man auf das Senden-Symbol unten rechts im Bildschirm (Siehe Abbildung 17). Die Mitteilung wird nun in der Firebase Datenbank gespeichert. Auf diese Änderung reagiert der Datenbank-Trigger "sendMessageNofitications" in den

Firebase Functions:

```
exports.sendMessageNotifications =
functions.database.ref('clubs/{club_id}/messages/{mes_id}').onCreate(async (snapshot, context)
(..)
```

Dieser Trigger wird ausgeführt, sobald ein neues Child Object unter "messages" im Club-Object erstellt wird. Der Firebase Server sendet nun mithilfe von OneSignal die Push-Notification an die ausgwählten Vereine und Gruppen. Für die jeweiligen Vereinsmitglieder erscheint nun eine neue Mitteilung auf dem Startbildschirm der App.

7. Push-Notifications

8. Datensicherheit

Einer der Gründe für das Nutzen der App sollte die Datensicherheit sein, welche von alternativen Anbietern nicht angeboten wird. Dieser Gedanke wird durch zwei Grundprinzipen umgesetzt. Erstens so wenig Daten wie möglich über Nutzer und Vereine zu sammeln, und zweitens die geringe Menge an Daten so sicher wie möglich zu speichern. Bei werbefinanzierten Prdukten wie zum Beispiel Facebook wäre diese Strategie undenkbar, da durch die gesammelten Information gezielter Anzeigen geschaltet werden können.

8.1 Anonymes Verwenden der App

Die App kann daher als Vereinsmitglied komplett anonym verwendet weren, also ohne Regrestierung mit E-Mail Adresse oder Telefonnummer. Dabei ist jedoch das Problem, dass die Nutzerdaten verlogen gehen, sobald die App wieder deinstalliert wird. Dieses Problem besteht vorallem für Vereinsbetreiber, die bei einer Neu-Installation der App keinen Zugriff mehr auf ihren Verein hätten. Daher muss bei der Erstellung eines Vereins zunächst ein herkömmliches Konto mit E-Mail Adresse und Passwort erstellt werden.

8.2 Firebase

Das Firebase Backend wird von Google selbst gesichert. Auf den Inhalt der Datenbank und auf die Firebase Functions lässt sich nur mit dem Firebase

Administrator Acconut zugreifen. Die Datenbank ist zudem durch die sogenannten "Realtime Database Rules" geschützt. Normalerweise kann jeder angemeldete Benutzer alle Einträge der Datenbank lesen und verändern. Für dieses Projekt ist eine solche Einstellung jedoch nicht ausreichend. Durch die Anoyme Regrestierung ist es möglich ohne jegliche Verifizierung auf die Datenbank zuzugreifen, was ein hohes Sicherheitsrisiko darstellen würde, wenn die Datenbank nicht ausreichend gesichert wäre. Die "Realtime Database Rules" für dieses Projekt sind beispielsweise so eingestellt, dass jeder Nutzer nur in sein eigenes Nutzer-Objekt schreiben kann. Zudem ist es, bis auf Bentutzernamen und Profilbild, nicht möglich Daten anderer Nutzer auszulesen. Auch die Daten von privaten Vereinen, können nicht eingesehen werden, sofern der Nutzer kein Mitglied dieses Vereins ist. Vereinsdaten können auch nur von Vereinsbetreibern geändert werden. Außerdem ist das Hochladen von größeren Datenmengen, wie zum Beispiel sehr langen Zeichenketten, nicht möglich. Die "Realtime Database Rules" werden im JSON-Format geschrieben und in Firebase gespeichert.

Hier ein Beispiel für eine Datenbank-Regel:

```
{
        "rules": {
                "users": {
                        "$uid": {
                                ".write": "$uid === auth.uid"
                        }
                }
        }
}
```

Auf das Benutzer-Obejekt werden nur Schreibzugriffe vom Benutzer selbst zugelassen. "uid" ist hierbei die ID des Benutzers.

8.3 Ende-Zu-Ende Verschlüsselung

Die Ende-Zu-Ende Verschlüsselung ist eines der Hauptbestandteile des Datenschutzskonzepts. In der App werden sowohl private Chat Nachrichten als

auch Mitteilungen, die ausschließlich an private Gruppen gerichtet sind, verschlüsselt übertragen. In der Praxis heißt dies, das der Sender die Nachricht verschlüsselt bevor sie in die Datenbank gespeichert wird. In der Datenbank selbst ist die Nachricht dann selbst für den Systemadministrator nicht entschlüsselbar, auch Zwischeninstanzen (Man-in-the-Middle-Angriff) die den Internetverkehr mitlesen können die Nachricht nicht entschlüsseln. Somit sind persönliche Daten und Chatverläufe auch bei einem direkten Angriff auf die Datenbank geschützt.

Für dieses Projekt würde das RSA-Verfahren verwendet. Das RSA-Verfahren ist ein asymmetrisches kryptographisches Verfahren, das zum Verschlüsseln von zum Beispiel Nachrichten verwendet werden kann.

8.4 Der Weg einer Chatnachricht

Genauer lässt sich das RSA-Verfahren anhand von Chatnachrichten erklären.

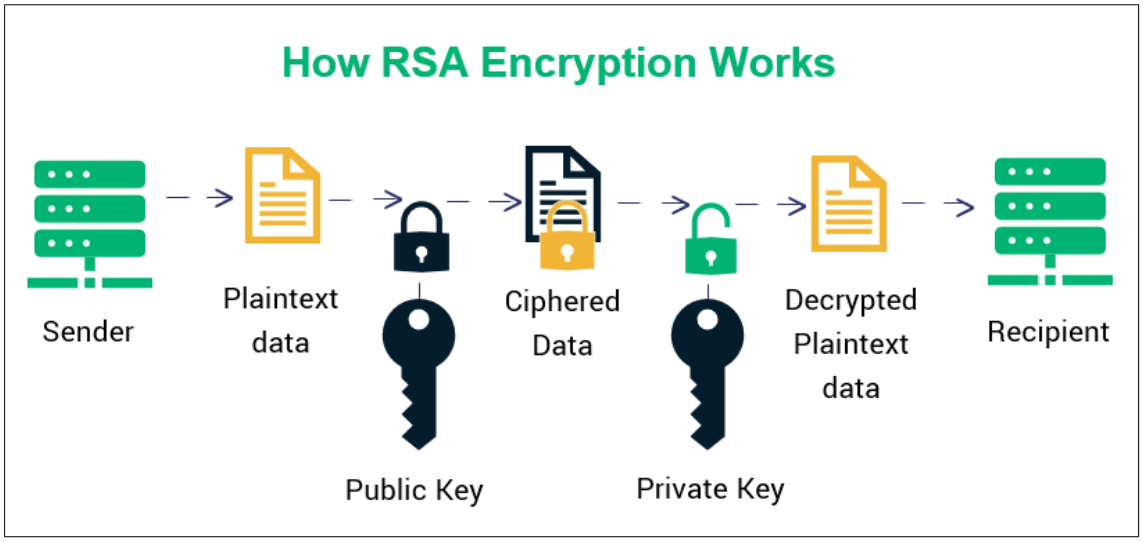


Abbildung 3

Wichtig für die Verschlüsselung nach dem RSA-Verfahren sind der öffentliche Schlüssel (Siehe Abb. 3, "Public Key") und der private Schlüssel (Siehe Abb. 3, "Private Key"). Dieses Schlüsselpaar wird bei der Einrichtung der App automatisch auf dem Gerät generiert. Der öffentliche Schlüssel wird dann für jeden öffentlich in der Datenbank gepsiechert. Mit diesem Schlüssel kann keine Nachricht entschlüsselt werden, er dient nur zur Verschlüsselung. Vergleichbar ist der öffentliche Schlüssel mit einem Schloss, welches Gegenstände zwar abschließen kann, aber ohne passenden Schlüssel nicht geöffent werden kann. Der private Schlüssel dient nun zur Entschlüsselung der Nachricht, also zum

aufschließen des Schlosses, dieser wird sicher auf dem Gerät des Eigentümers gespeichert. Der private Schlüssel kann, anders als bei einem Schloss, nicht vom öffentlichen Schlüssel hergeleitet werden.

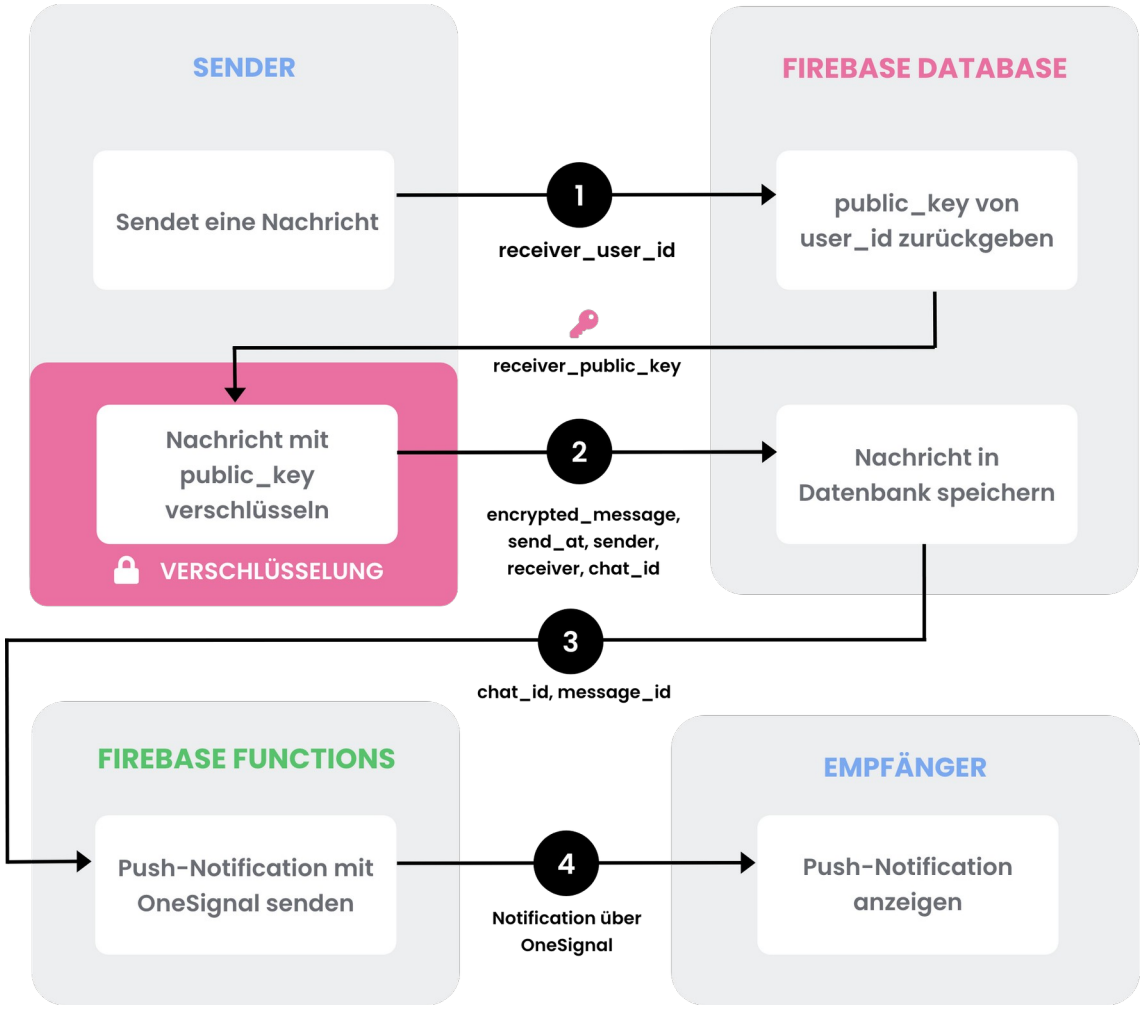
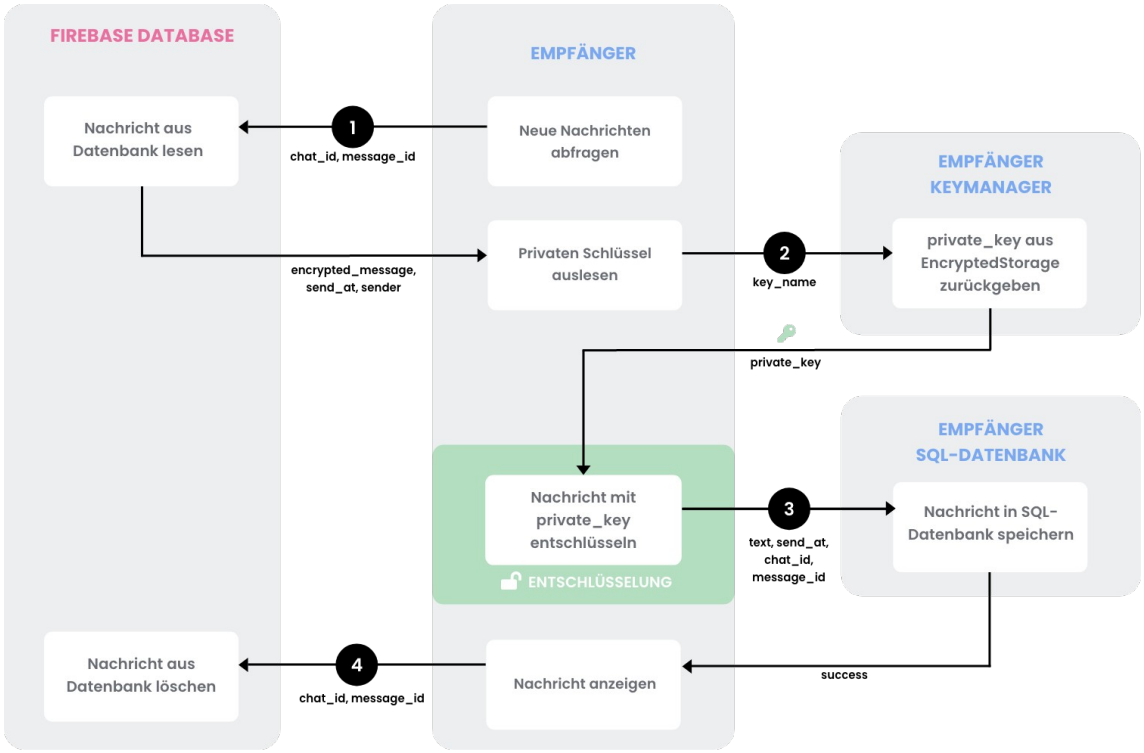


Abbildung 4

Wird also jetzt von Benutzer 1 eine Nachricht an Benutzer 2 gesendet, so muss der Sender zunächst den öffentlichen Schlüssel des Empfängers aus der Firebase Datenbank abfragen (Siehe Abbildung 4, Übergang 1). Der öffentliche Schlüssel für den jeweiligen Benutzer ist in der Firebase Datenbank unter "users/[user_id]/public_key' gespeichert. Die noch lesebare Nachricht wird dann mit dem öffentlichen Schlüssel des Emfpängers verschlüsselt (Siehe Abbildung 4, "Nachricht mit public_key verschlüsseln"). Benutzer 1 läd nun die verschlüsselte Nachricht in die Firebase Datenbank hoch (Siehe Abbildung 4, Übergang 2). Die Methode "sendChatMessageNotifications" in den Firebase Functions reagiert nun auf die hochgeladene Nachricht und sendet per OneSignal eine Push-Notification an den Emfpänger der Nachricht (Siehe Abbildung 4, Übergang 3 und 4).


FIREBASE DATABASE
EMPFÄNGER
Nachricht aus Datenbank lesen
1
chat_id, message_id
Neue Nachrichten abfragen
EMPFÄNGER KEYMANAGER
encrypted_message, send_at, sender
Privaten Schlüssel auslesen
2
key_name
private_key aus EncryptedStorage zurückgeben
private_key
EMPFÄNGER SQL-DATENBANK
Nachricht mit private_key entschlüsseln
ENTSCHLÜSSELUNG
3
text, send_at, chat_id, message_id
Nachricht in SQL-Datenbank speichern
Nachricht aus Datenbank löschen
4
chat_id, message_id
Nachricht anzeigen
success

9. Besondere Stellen im Code

9.1 DatabaseConnector.js

9.2 HeaderScrollView.js

9.3 ScreenHandler.js

Abbildungsverzeichnis

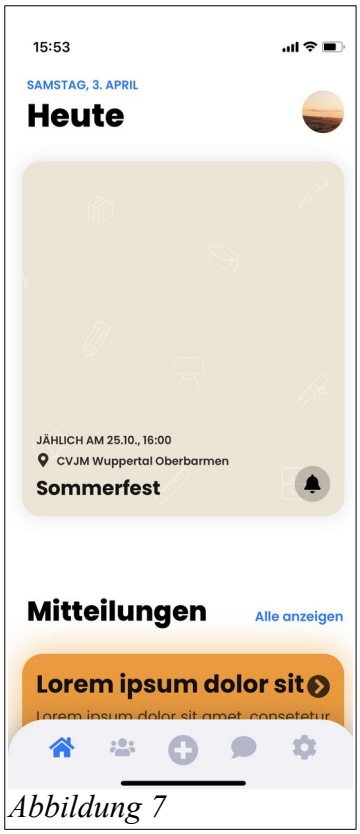


Abbildung 7

Abbildung 5

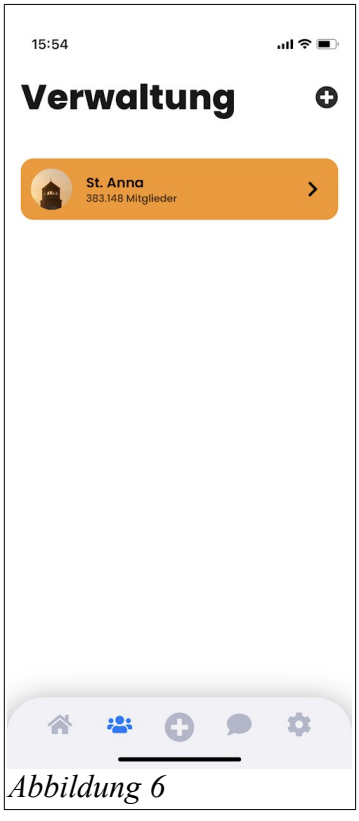


Abbildung 6

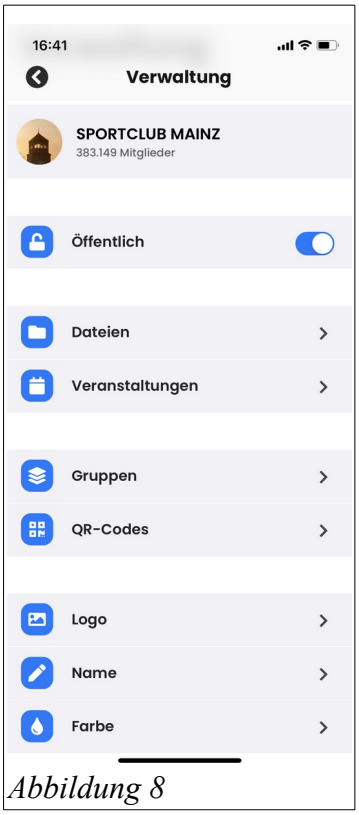


Abbildung 8

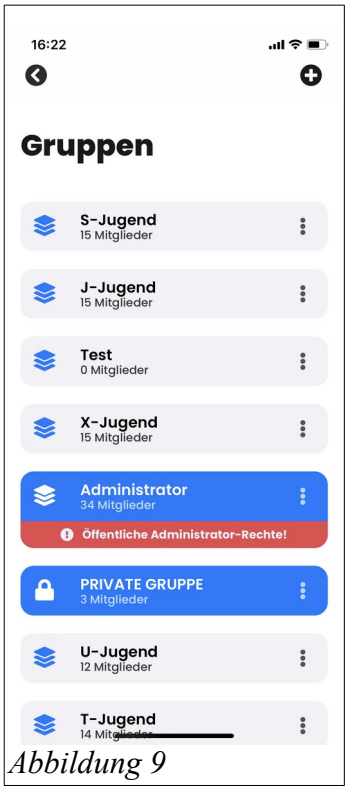


Abbildung 9

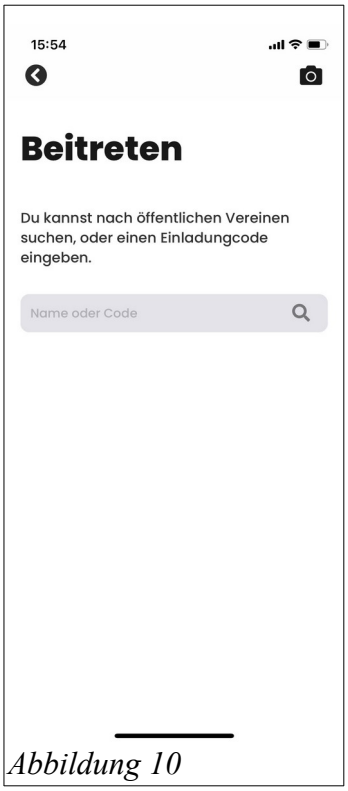


Abbildung 10

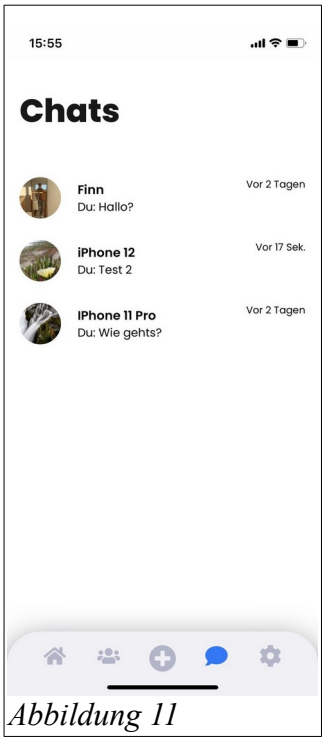


Abbildung 11

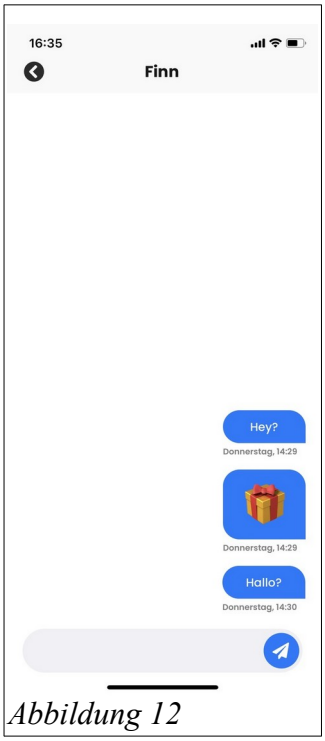


Abbildung 12

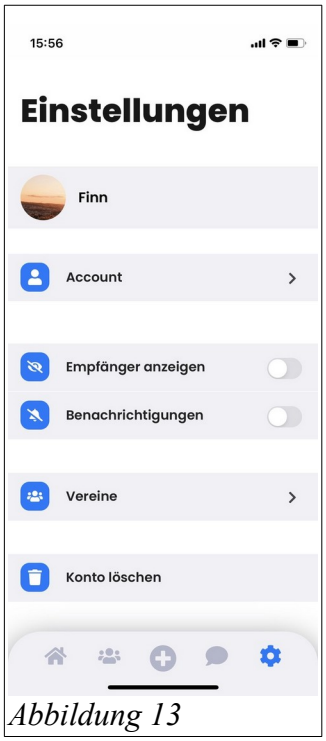


Abbildung 13

Abbildung 14

Abbildung 15

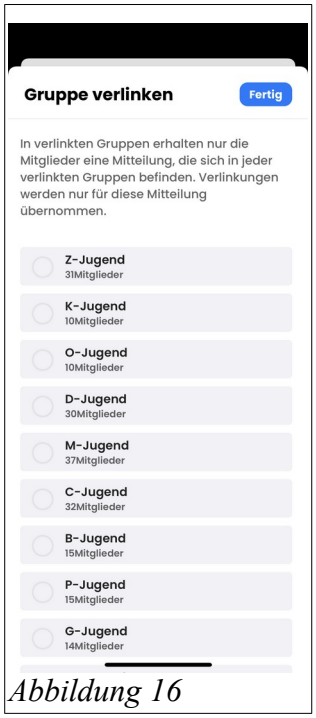


Abbildung 16

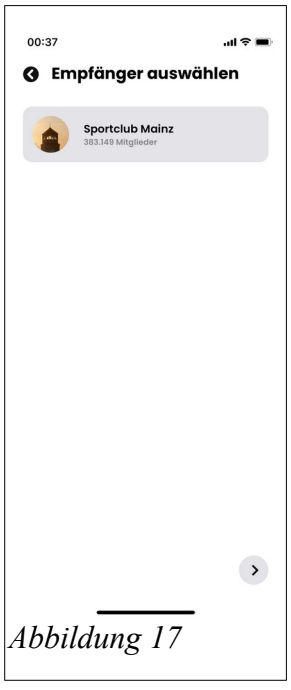


Abbildung 17

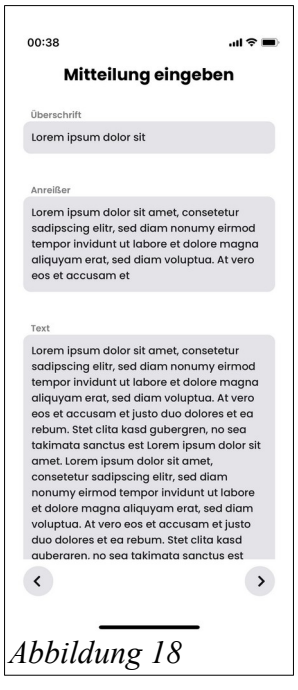


Abbildung 18

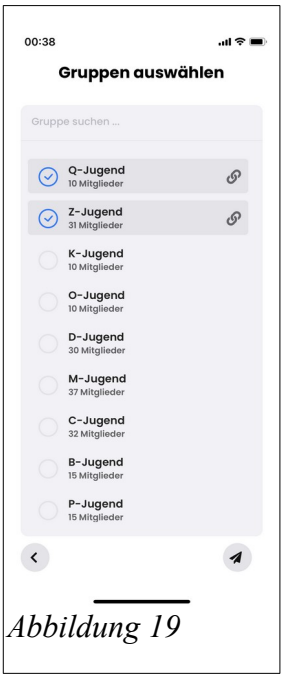


Abbildung 19

Literaturverzeichnis

- FIREBASE – INSTALLATION & SETUP IN JAVASCRIPT (2020): https://firebase.google.com/docs/database/web/start – Zugriff: 22.04.2020

- FIREBASE - A COMPREHENSIVE APP DEVELOPMENT PLATFORM (2020): https://firebase.google.com/ - Zugriff: 22.04.2020

- FIREBASE - UNDERSTAND FIREBASE REALTIME DATABASE RULES (2020): https://firebase.google.com/docs/database/security – Zugriff: 22.04.2020

https://cloud.google.com/storage/docs/locations?hl=de#available-locations