egoraDokumentation

Eine Android-App für den Verleih in deiner Nachbarschaft

31. Juli 19

Projekt C

Media Systems (B.Sc.)

Hochschule für angewandte Wissenschaften Hamburg /
Hamburg University of Applied Sciences

Department Medientechnik
Fakultät Design, Medien und Information

Dominik Stecher 2288508, Nina Tietje 2238710

Prüfer: Prof. Dr.-Ing. Sabine Schumann

Inhaltsverzeichnis

1. Vorstellung des Projektes

Für unser Modul Projekt C an der HAW Hamburg erstellten wir die Android Anwendung egora, mit der Nachbarn und Gemeinschaften eine Möglichkeit bekommen sollen, Gegenstände kostenfrei untereinander zu verleihen. Durch die App soll die Kommunikation und Organisation für den Verleih vereinfacht und verbessert werden.

1.1 Idee

Ideengeber für unser Projekt war unsere Professorin und Betreuerin Sabine Schumann. Sie veranschaulichte bei der Auftaktveranstaltung des Wintersemesters 2018 an der HAW das Problem des Mietshauses, in dem sie wohnt. Dort führt die Nachbarschaft eine schriftliche Liste, in der die Anwohner Nutzgegenstände eintragen können, die man gelegentlich benötigt, wie z.B. Akkubohrer oder Luftmatratze. Diese Liste wurde aber schnell unübersichtlich durch Streichungen von Gegenständen und Platzmangel auf der Liste. Es lag daher nahe, dieses Problem durch eine digitale Anwendung zu lösen.

1.2 Zielsetzung

Ziel war es, eine Softwarelösung für das eben dargestellte Problem zu erstellen. Die Anwendung soll auf Smartphones und Tablets nutzbar sein. Die Vorteile von mobilen Geräten sind die hohe Verbreitung und der schnelle Zugriff. So kann der Nutzer im Fall, dass er gerade einen bestimmten Gegenstand benötigt, direkt auf seinem Smartphone danach suchen. Dadurch ist eine geringe Einstiegshürde gegeben und hoher Nutzerkomfort.

Da die Gruppe selbst Android-Geräte besitzt und diese häufig als Smartphones genutzt werden, entschieden wir uns, die Anwendung nur für das Android-Betriebssystem zu programmieren. Durch den Fokus auf nur ein Betriebssystem lässt sich der Programmieraufwand reduzieren und es bleibt mehr Raum für die Optimierung der Usability und Performance.

Weiterhin soll die App simpel gehalten werden. Dies bedeutet, dass wir nur die Funktionen einbauen wollten, die für einen reibungslosen Verleih unter Nachbarn benötigt werden. So blieb die Bedienung auf einem selbsterklärenden Niveau.

2. Konzeption

Zu Beginn der Planung formulierten wir alle Module aus, die für die App benötigt werden. Zunächst muss der Nutzer sich einen persönlichen Account anlegen können und sich mit diesem anmelden können. Im Weiteren muss der Anwender dazu in der Lage sein, sich mit seiner Nachbarschaft bzw. Community vernetzen zu können. Innerhalb der Community müssen Items eingestellt, eingesehen und gelöscht werden können. Sollten sich Informationen des persönlichen Profils ändern, so sollte der Nutzer keinen neuen Account anlegen müssen. Um sich über die Gegenstände auszutauschen und einen möglichen Abholtermin zu vereinbaren, soll der Anwender innerhalb der App Chatnachrichten verfassen und empfangen können. Aus diesen und weiteren User Stories formulierten wir fünf Module; Anmeldung und Registrierung, Communities, Items, Profil und Chat.

Im nächsten Schritt legten wir für die einzelnen Funktionalitäten die nötigen Views fest und skizzierten erste Layoutentwürfe.

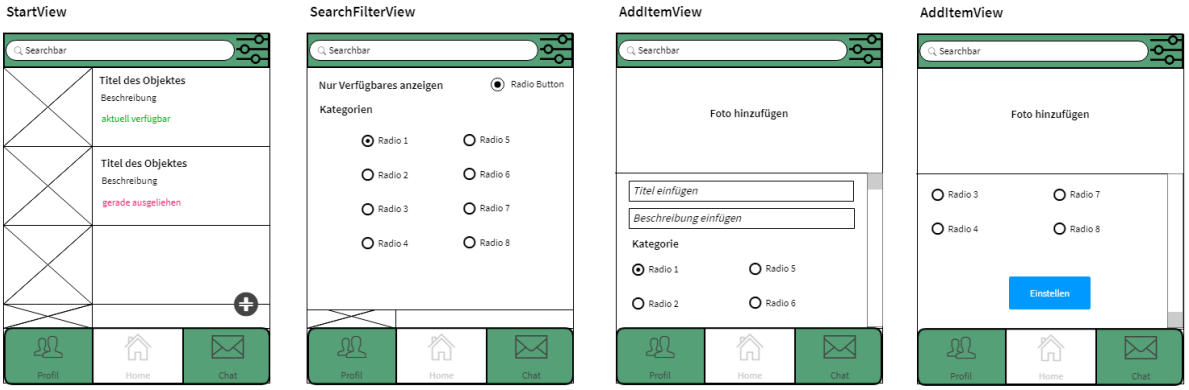


Abbildung 1: Mockup-Skizzen

Um unsere Dokumente zu verwalten und einzusehen, nutzten wir einen gemeinsamen Cloudspeicher. Wir entschieden uns, beim Projekt einem inkrementellen Vorgehensmodell zu folgen und gliederten daher die einzelnen Module in kleinere Arbeitspakete und wiesen diese den Gruppenmitgliedern zu. Durch die Aufteilung des Projektes in kleine Teile kann zum einen der Entwicklungsstand des Projektes vom gesamten Team besser nachvollzogen werden und es vereinfacht die Teamkommunikation bei Problemen. Hierbei nutzten wir *Github Projects*. Nachdem das Projekt umrissen war, machten wir uns an die Programmierung der einzelnen Module.

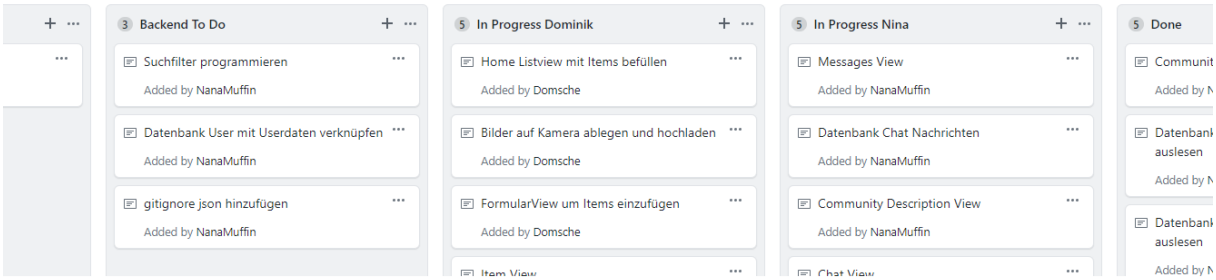


Abbildung 2: Github-Projects

3. Umsetzung

Die App egora wurde für Android API 23 bis 28 konzipiert und entwickelt, das bedeutet, Android 6.0 bis Android 9.0 werden unterstützt.

3.1 Software und Frameworks

Android Studio

Egora wurde ausschließlich über die Entwicklungsumgebung Android Studio[1] entwickelt. Hierbei wurden die Android-Activities in Java und die Layouts mit XML programmiert.

Google Firebase

Durch die gute Anbindung in Android Studio und eine attraktive Kostenstruktur entschieden wir uns, Google Firebase als Entwicklungsplattform für unsere Anwendung zu nutzen. Durch

[1] Android Studio - Android Entwicklungsumgebung, siehe Quellenverzeichnis

Firebase können komplexe Abfragen mit niedrigem Coding-Aufwand erzeugt sowie sehr schnell verarbeitet und weitergeleitet werden. Firebase ist eine NoSQL Datenbank inkl. Datenbankserver und nicht relational. Für die Speicherung der Artikelbilder nutzen wir den von Firebase bereitgestellten Firebase Storage[2].

Zusätzlich verwenden wir den Cloud-Speicherservice Firestore[3], in diesem werden fast alle Daten der egora App gespeichert. Mit Firestore kann man sehr einfach Daten mittels Queries

abfragen, filtern und sortieren. Außerdem können die Informationen ganzer Klassen im Firestore gespeichert werden. Hierbei werden sie zwar für Firestore umkonvertiert, können aber beim Abfragen der Daten recht simpel wieder in die Ursprungsklasse umgewandelt werden. Dafür benötigt die Klasse einen Konstruktor ohne Parameter und jede Variable, die abgespeichert und aufgerufen werden soll, benötigt eine Getter- und Setter-Methode.

Firebase bietet zudem eine eigene Datenbank zur Verwaltung der Anmeldung von Nutzern. Dieses Modul nennt sich Firebase Authentication[4] und bietet verschiedene Dienste für das Back-End an. So können Nutzer hierüber z.B. eingeloggt, registriert und auf den Login-Status geprüft werden. Über definierte Regeln innerhalb der Datenbanken wird gewährleistet, dass nur registrierte Nutzer freien Zugriff auf die Anwendung und die Daten aus dem Backend bekommen. Diese Regeln kann jeder Entwickler für seine Anwendung selbst optimieren.

FirebaseRecyclerAdapter

Um unsere Listen in den Modulen Items, Profil und Chats darzustellen und zu aktualisieren, nutzten wir unter anderem den FirebaseRecyclerAdapter. Um diesem die zugehörigen Informationen zu übergeben, wird zuvor eine Suchanfrage an die Datenbank gestellt. Die erhaltenen Daten werden dann vom Adapter wieder zurück in die (selbst) angegebene Klasse umgewandelt, die zur Speicherung im Firestore verwendet wurde. Die Informationen jedes Treffers werden innerhalb des Adapters an sog. Holder gebunden und dieser gibt die Anzahl und das Layout der Spalten vor. Besonders praktisch ist hierbei, dass der Adapter sich selbstständig updatet, sobald eine Änderung in der Datenbank getätigt worden ist.

3.1.1 Libraries

FancyToast-Android

Um dem Nutzer Mitteilungen in der App anzuzeigen, verwenden wir die Fancy Toast-Android Library[5]. Diese stellt eine Toast-Klasse zur Verfügung mit vorgefertigten UI Layouts, beispielsweise für Error- und Success-Meldungen.

Pricasso

Da die Bildrahmen innerhalb der App unterschiedliche Formate benötigten, mussten wir eine Möglichkeit finden, um die Bilder in ein einheitliches Bildformat auszurichten und zu skalieren. Hierzu nutzten wir Picasso[6], welches mit wenigen Befehlen das Bild beschneidet, an den Rahmen anpasst und abschließend die ImageView befüllt. So konnten wir auch bei unterschiedlichen Fenstergrößen die Bilder von Objekten in allen Activities uniform benutzen.

---

[2] Firebase - Firestore Doku, siehe Quellenverzeichnis
[3] Firebase - Storage Doku, siehe Quellenverzeichnis
[4] Firebase - Authentication Doku, siehe Quellenverzeichnis
[5] FancyToast-Android Library, siehe Quellenverzeichnis
[6] Picasso - Android Library, siehe Quellenverzeichnis

3.2 Datenbankstruktur

Die meisten unserer Daten werden im Firestore gespeichert. Dabei werden die Daten in Sammlungen abgelegt, diese werden als Collection bezeichnet. In einer Collection kann man Dokumente abspeichern oder auch weitere Collections (Subcollections), die wiederum Dokumente enthalten können. In Abbildung 3 sieht man einen Ausschnitt der Firestore Cloud. Links sind die Collections, in der Mitte die Dokumentenliste und rechts die Daten des ausgewählte Dokument.

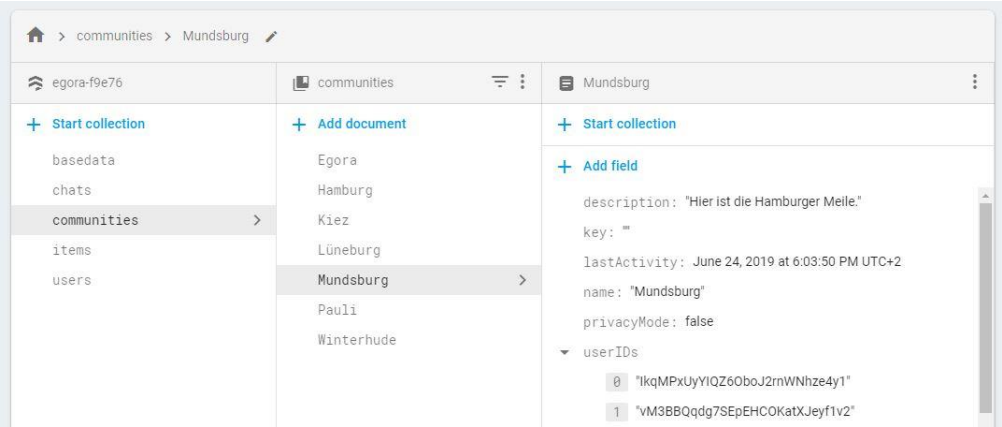
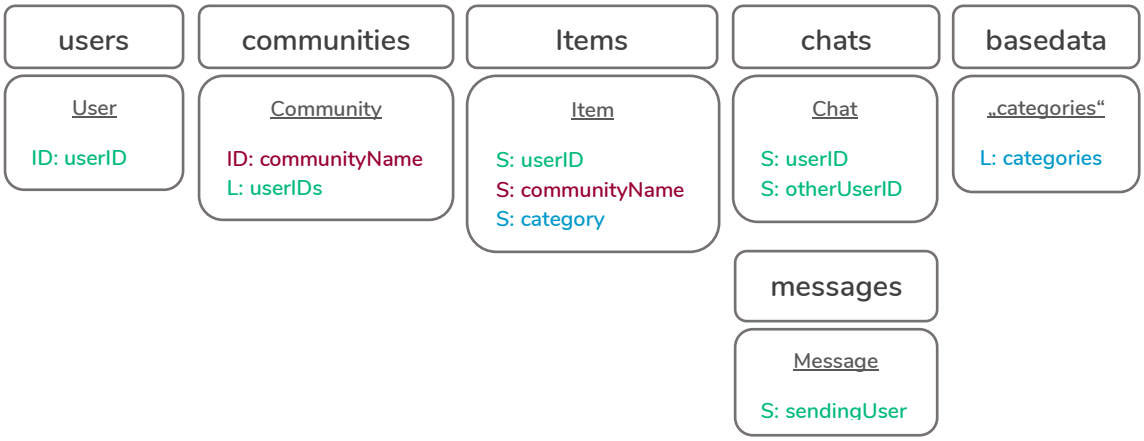


Abbildung 3: Firestore Cloud

In der unteren Infografik sind unsere Collections in Firebase aufgeführt, *user*, *communities*, *items*, *chats* und *basedata*. Die darunter folgenden Boxen veranschaulichen den Typ, der darin abgelegten Dokumente. In der *chats* Collection werden z.B. Chat-Dokumente gespeichert, die wiederum eine Subcollection *messages* besitzen, wo die Nachrichten des Chats abgelegt werden. Die Variablen, die in den Dokumenten angegeben werden, zeigen die Datenverknüpfung zwischen den Collections. Steht *ID* vor einer Variable, bedeutet dies, dass die Variable nicht Inhalt des Dokumentes ist, sondern die Dokumenten-ID selbst. *S* steht für einen String und *L* für eine Liste oder ein Array. In der Collection *basedata* ist nur ein einziges Dokument namens *categories* abgespeichert, welches eine Liste mit Kategorienamen der Items beinhaltet.



3.3 Module

Zunächst sollen die einzelnen Module näher erklärt werden. Hierbei gehen wir sowohl auf das Zusammenspiel der einzelnen Klassen miteinander ein als auch auf den jeweiligen Umfang der Module. Die Hauptfunktionen der App sind nur zugänglich, sofern der Login zuvor erfolgreich durchgeführt wurde.

3.3.1 Anmeldung und Registrierung

Nachdem die App gestartet wird, gelangt der Nutzer auf die Login-View. Hier kann er sich durch Eingabe seiner registrierten E-Mail und seines Passworts anmelden und gelangt so in den geschützten Bereich der App. Sobald der Nutzer seine Eingabe über den Login-Button bestätigt, durchsucht die Anwendung die Datenbank nach einem passenden Eintrag zur E-Mail des Nutzers und gleicht anschließend die Passwörter ab.

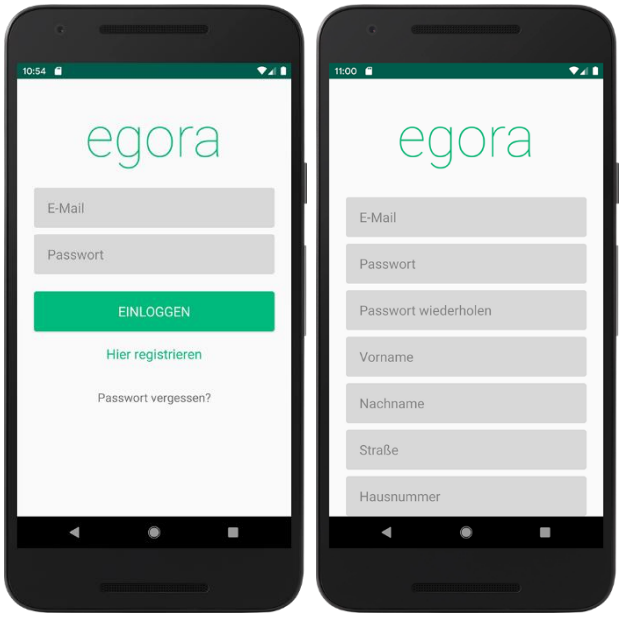


Abbildung 4: Login View und Register View, v.l.n.r.

Wird die App zum ersten Mal genutzt, so muss vorher noch ein Account angelegt werden. Das Formular zur Registrierung kann vom Login-View aus angesteuert werden. Es besteht aus mehreren Feldern, in die der Nutzer die persönlichen Daten hinterlegt. Sind alle Felder ausgefüllt, so kann der Upload-Prozess gestartet werden und die Informationen werden in der Datenbank abgelegt. Die meisten Daten des Nutzers können im Nachhinein noch angepasst werden. Einsicht in Adresse und Namen besitzen nur Mitglieder aus der Community, in der sich auch der Nutzer selbst befindet. Weitere Daten werden nicht geteilt.

3.3.2 Communities

Damit die Anwender sich mit ihren Freunden und Bekannten vernetzen können, werden in egora Communities genutzt. Um Mitglied einer Community zu werden, kann der Nutzer entweder selbst eine erstellen oder einer bestehenden beitreten.

Nach der Registrierung wird die Communities-Übersicht angezeigt. Hier wird eine Liste mit den bisher hinzugefügten Communities aus der Datenbank geladen, die man mit dem oberen Suchfeld durchsuchen kann. Die Suche filtert nach teilweiser bis ganzer Übereinstimmung des Suchbegriffs mit dem Artikelnamen.

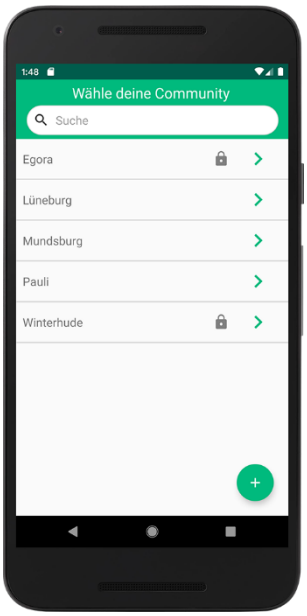


Abbildung 5: Communities-Übersicht

Wählt man eine Artikelzeile aus der Liste aus, werden Name und Beschreibung der Community angezeigt. Ist die Community privat, muss man im unteren Feld erst den richtigen Zugangscode eingeben, um beizutreten (siehe Abb. 7). Wenn man selbst eine Community erstellt, wird man ihr auch automatisch hinzugefügt. Bei der Erstellung einer Community muss man einen einmaligen Communitynamen, zur Abspeicherung in der Datenbank, angeben. Damit ist gewährleistet, dass ein Nutzer bei der Suche nach einer bestimmten Community nur ein Resultat erhält und keine Verwechslung auftreten kann. Eine Beschreibung und ein Zugangscode für eine private Community können auch angegeben werden, sind aber optional.

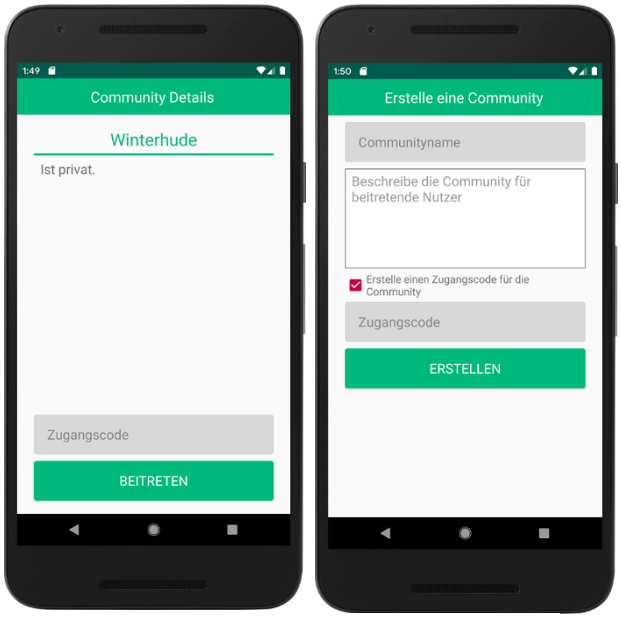


Abbildung 6: Communities-Detailansicht und Community-hinzufügen-View, v.l.n.r.

Die Communities werden in Firestore in der Collection communities gespeichert. Es gibt eine Community Klasse, die alle Community Details und eine Liste beinhaltet, in denen die User-IDs der Mitglieder gespeichert werden. Die Community-Klasse wird auch zur Speicherung der Daten im Firestore verwendet. Bisher können die Daten einer Community nicht im Nachhinein vom Nutzer bearbeitet werden und die Community kann auch nicht gelöscht werden.

3.3.3 Items

Das Modul zur Verwaltung von Items umfasst drei unterschiedliche Activities. Hierbei werden im Home-Bereich der Anwendung alle Gegenstände angezeigt, die den Teilnehmern der jeweiligen Community gehören. Diese können über eine Filterfunktion mit vorgegebenen Kategorien und der Titelsuche eingegrenzt werden. Die Titelsuche funktioniert wie eine Wildcard-Suche und filtert während des Schreibens bereits die passenden Artikel heraus. Die Nutzer können auch eine Kategorie durchstöbern, ohne einen Suchbegriff dazu einzugeben.

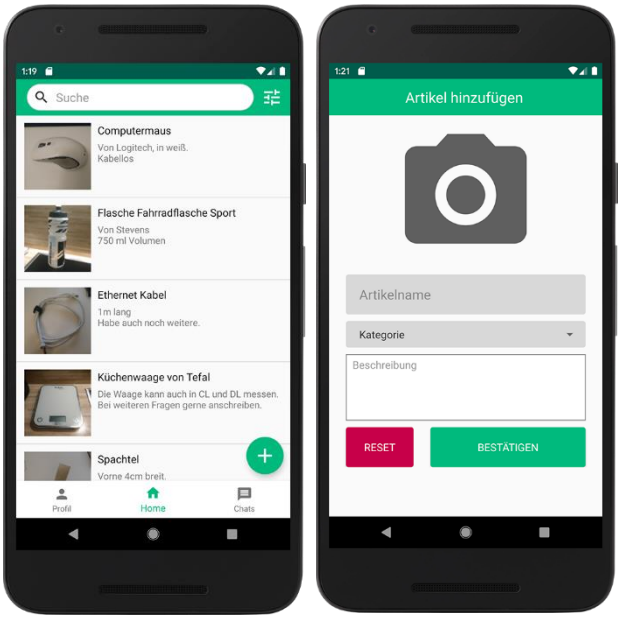


Abbildung 7: Home-View und Item-hinzufügen-View, v.l.n.r.

Damit Gegenstände in die Datenbank gelangen, müssen diese über eine eigene Activity hinzugefügt werden. Um hierhin zu gelangen kann der Nutzer den + Button im Home-Bereich betätigen. Es öffnet sich ein Formular, siehe Abb. 7 rechts. Durch das Anklicken des Kamera-Icons startet die Kamerafunktion des Smartphones. Nach der Aufnahme des Bildes

wird die erzeugte Bitmap an die Anwendung weitergegeben und im Bildrahmen angezeigt. Sind der Name des Gegenstandes, eine kleine Beschreibung (optional) und die Kategorie ausgefüllt, kann das Formular abgeschickt werden. Die Kategorien sind vorgegeben und umfassen die sieben Bereiche: Basteln, Elektronik, Garten & Grill, Handwerken, Küche & Haushalt, Sonstiges und Sport & Freizeit. Bild, Name und Kategorie sind Voraussetzung, um einen Artikel hinzuzufügen.

Sollten die Eingaben fehlerhaft sein, so kann der Nutzer über den Reset-Button alle Einträge zurücksetzen. Die schriftlichen Infos des Gegenstandes werden nach Bestätigung in der Datenbank abgespeichert inkl. der User-ID des Besitzers bzw. Erstellers, um später die nötigen Daten auslesen zu können. Das Bild des Gegenstandes wird im Firebase Storage abgelegt, da Bilder nicht direkt im Firestore gespeichert werden können.

Abbildung 8: Item-Detailansicht der Item-Activity

Der letzte Teil des Moduls ist die Darstellung der gesamten Informationen eines Gegenstandes in übersichtlicher Form (rechtes Bild). Hierbei wird über das Anklicken einer Gegenstandszeile im Home-Bereich die sog. Item-Activity gestartet. Dort wird der Gegenstand mit vergrößertem Bild und anschaulichen Textfeldern angezeigt. Sollte nun der Nutzer Kontakt mit dem Besitzer aufnehmen wollen, so kann er hier über *Besitzer anschreiben* ein Chatfenster mit dem anderen Anwender öffnen.

3.3.4 Profil

Um eine Übersicht über die eigenen Informationen zu erhalten, wurde das Profil-Modul erstellt. Hier kann der Nutzer neben den Gegenständen, die er zum Teilen freigegeben hat, auch seine hinterlegten Daten einsehen und ändern bzw. löschen. Daher teilt sich das Modul in die Bereiche *Meine Anzeigen* und *Benutzerkonto*. Die Trennung äußert sich sowohl visuell über zwei unterschiedliche Tabs als auch im Code über zwei eigenständige Fragmente.

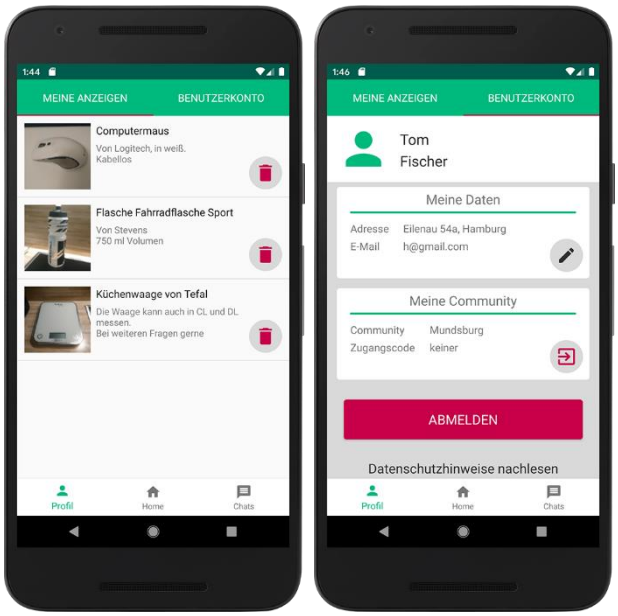


Abbildung 9: Meine Anzeigen View und Benutzerkonto View, v.l.n.r.

Im Tab der eigenen Gegenstände werden diese in Listenform durch den FirebaseRecyclerAdapter aufgeführt. Neben Bild und Details enthalten die Listeneinträge auch ein Papierkorb-Icon, welches zum Löschen dient. Wird dieses angeklickt, öffnet die App einen Alarm-Dialog, der abfragt, ob sich der Nutzer sicher ist, den Löschvorgang fortsetzen zu wollen. Nach Bestätigung werden alle Informationen des Gegenstandes aus den Datenbanken gelöscht.

Damit der Nutzer seine persönlichen Daten prüfen und ggf. ändern kann, muss er sich in den *Benutzerkonto* Bereich begeben. Dort befinden sich neben den Informationen ein Button zum Ändern von Namen, Adresse und Passwort , sowie ein Button, um die Community zu wechseln. Bei der Änderung der Daten muss ein Formular ausgefüllt werden, in das die zu aktualisierenden Werte eingetragen werden müssen. Wird ein Feld nicht ausgefüllt, bleibt der Eintrag in der Datenbank unverändert. Scrollt man nach ganz unten, findet man einen Link, über den man die Datenschutzhinweise einsehen kann, die Anzeige erfolgt mit dem Android PDF Viewer[7].
Beim Klick auf *Wechsle Community* erzeugt die App einen Alarm-Dialog, in dem der Vorgang erneut bestätigt werden muss. Anschließend gelangt man ins Community-Modul, wo eine neue Community ausgewählt werden kann. Während des Wechsels werden alle Gegenstände in einen temporären *changing*-Status in der Datenbank gesetzt. Sobald man erfolgreich einer neuen Community beigetreten ist, wird der Status von *changing* auf den Namen der neuen Community gesetzt. Dabei ziehen auch alle Gegenstände des Nutzers mit in die neue Community um.

3.3.5 Messenger

Der Messenger ist in der App unter dem Menüpunkt *Chats* zu finden. Er beinhaltet zwei Activities, die eine zeigt alle Chats des Nutzers, die andere öffnet sich bei der Auswahl eines Chats aus der Liste und zeigt den Chatverlauf mit dem Besitzer des jeweiligen Gegenstandes. Auch im Messenger wird der FirestoreRecycler-Adapter verwendet.

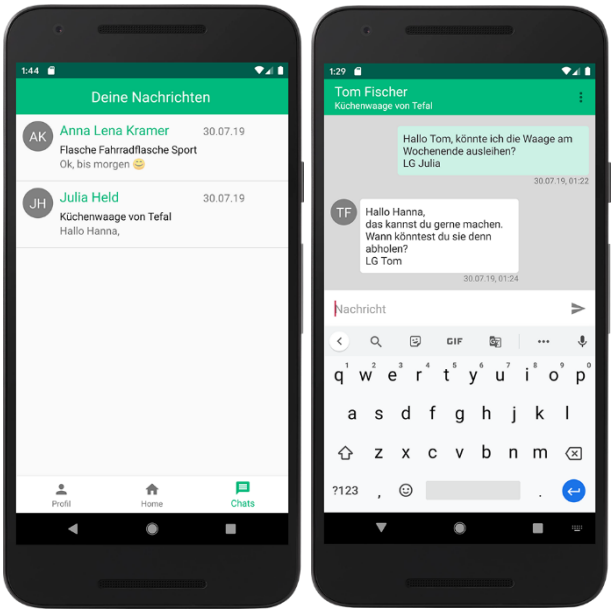


Abbildung 10: Messenger Übersicht und Chatverlauf, v.l.n.r.

Die Chats in der Übersicht des Messengers werden nach dem Datum der letzten Nachricht sortiert. Man kann auch mehrere Chats mit dem gleichen User haben, was sinnvoll ist, um Unterhaltungen zu verschiedenen Gegenständen zu führen.

Einen Chat kann man nur starten, indem man über die Artikelseite dem Besitzer schreibt. Klickt man auf den *Besitzer kontaktieren*-Button, öffnet sich das Chatfenster (rechtes Bild). Dabei werden auch die Chatdokumente in der Datenbank angelegt. Damit jeder User den Chat für sich löschen kann, gibt es je User ein Chatobjekt (lokal) bzw. Chatdokument (in der Datenbank). Gleich sind dabei die Messages, die in einer Subcollection unter dem Chat-Dokument im Firestore gespeichert werden. Jedes Mal, wenn das Senden-Icon geklickt wird, wird auf die Chatdokumente beider Nutzer zugegriffen und die Message unter der Collection *messages* hinzugefügt. Um einen Chat zu löschen, muss man über das obere Drei-Punkte-Menü *Chat löschen* auswählen und einen Alarm-Dialog bestätigen. Dadurch wird das eigene Chatdokument in der Datenbank gelöscht inkl. der *messages* Subcollection und der Chat

[7] Android PDF Viewer - Library

verschwindet in der Messenger-Übersicht. Der Chatpartner behält seinen Chat weiterhin und kann auch wieder darin zurückschreiben. Die Nachrichten im Chatverlauf werden nach der Uhrzeit und dem Datum geordnet, die auf dem erstellenden Android Gerät eingestellt sind.

6. Projektumfang

Arbeitsschritt	Dominik Stecher	Nina Tietje
Planungs-Recherche	20 h	15 h
Planung & Konzeption	25 h	25 h
Firebase Installation & Verwaltung	10 h	10 h
Backend Development	70 h	55 h
Datenbank Development	40 h	30 h
UI Layouts	20 h	50 h
Development Recherche	65 h	60 h
Debuggen	70 h	60 h
Dokumentation & Handbuch	20 h	18 h
Gesamt	**340 h**	**323 h**

Tabelle 1: Zeitplan in Stunden, nach Arbeitsbereichen aufgeteilt

Modul	Bereich	Dominik Stecher	Nina Tietje
Login & Registrierung	Backend/Datenbank UI	90 % 15 %	10 % 85 %
Community	Backend/Datenbank UI	10 %	90 % 100 %
Items	Backend/Datenbank UI	75 % 40 %	25 % 60 %
Profil	Backend/Datenbank UI	90 % 40 %	10 % 60 %
Messenger	Backend/Datenbank UI	10 %	90 % 100 %
Datensicherheit		100 %	

Tabelle 2: Arbeitsaufteilung nach Modulen

5. Problemstellungen

Firebase

Als wir schon mitten in der Umsetzung des Projektes waren, wurden wir darauf aufmerksam, dass neben der Firebase Database auch die Firestore Cloud zur Datenspeicherung unserer App genutzt werden könnte. Da wir zu dem Zeitpunkt Probleme hatten, bei unserer Datenabfrage mehrere Datensätze nach bestimmten Informationen zu durchsuchen, sind wir dann auf Firestore umgestiegen. Firestore bietet wesentlich mehr Möglichkeiten an, um mit Queries zu arbeiten.

Android Studio

Außerdem haben wir recht schnell angefangen, mit Activities in Android Studio zu entwickeln, da es bisher die bekannteste und simpelste Art ist, um Apps in Android zu programmieren. Wir haben erst zum Ende der Entwicklung festgestellt, dass wir vor allem für einen guten und schönen Übergang bei einer Bottom-Navigation auf Fragments für Home-, Profil- und Messenger-View hätten zurückgreifen müssen, anstatt auf Activites. Die Änderung dieser Views in Fragmente konnte während dieses Projektes leider nicht mehr umgesetzt werden.

Picasso

Nutzt man die Picasso-Library, so lässt sich zwar schnell ein Bild an eine ImageView übergeben, aber hierbei wird das Bildformat der originalen Bilddatei an die ImageView angepasst. Um dies zu umgehen, nutzten wir eine Ausschneidefunktion der Library, durch die das Bild skaliert wird und anschließend die überschüssigen Ränder abgeschnitten. So wurden die Bilder nicht mehr verzerrt und hatten ein einheitliches Format.

Chat

Bei der Konzeption der Chats bestand die Frage, ob man einen Chat für zwei User erzeugt oder einen je Nutzer. Wir wollten das Löschen eines Chats vom User *A* ermöglichen, ohne dass dieser für den Chatpartner *B* verschwindet. Außerdem sollten die (vor dem Löschen) gesendeten Nachrichten nicht wieder bei User *A* erscheinen, wenn der Chatpartner *B* im Chat eine neue Nachricht an User *A* sendet. Das hätte bei einem einzelnen Chatobjekt für zwei Personen nur mit Booleans für alle Messages oder einem Zeitstempel ab Löschung funktioniert. Da wir keinen begrenzten Speicherplatz bei Firestore haben und die Schnelligkeit der Abfrage durch Queries unabhängig von der gespeicherten Datenmenge ist, gibt es für uns keine merkbaren Nachteile, gleiche Daten mehrfach (zwei Chatobjekte) in Firestore abzuspeichern. Daher haben wir uns dafür entschieden, ein Chatobjekt je Nutzer anzulegen. Das empfanden wir als einfacher und zugleich ausbaufähiger, um z.B. später einzelne Nachrichten löschen zu können. Denn sonst müsste jede Message für jeden Nutzer einen Boolean beinhalten, der sagt, ob sie angezeigt wird oder nicht.

Bei der Entwicklung des Chats kam es auch noch zu einem anderen Problem. Objekte der Klasse Message konnten zwar in einer Liste des Chatobjektes gespeichert und so auch in den Firestore übertragen werden, sie konnten aber nicht zurück in die Message Klasse umgewandelt werden, was Probleme beim Lesen der Daten verursachte. Man hätte zwar auch ohne den Objektumwandlung an die Daten kommen können, aber auf einem umständlicheren Weg. Daher speichern wir die Messages nun nicht mehr im Chatobjekt (lokal) und damit im Chatdokument (Datenbank), sondern in einer Subcollection des Chatdokumentes.

Die Messages, die man im Chat sendet, werden mit einem Zeitstempel versehen. Er wird benötigt, um die Nachrichten später in der richtigen Reihenfolge anzuzeigen. Firestore selbst speichert die Dokumente beim Hinzufügen nicht (immer) in der Erstellungsreihenfolge. Der Zeitstempel wird erfasst, indem die Systemzeit auf dem Smartphone abgefragt wird. Hat einer der Chatteilnehmer jedoch nicht die automatische Uhrzeit- und Datumserfassung eingeschaltet, werden die Nachrichten dementsprechend in falscher Reihenfolge angezeigt. Die Zeitzone hat darauf keinen Einfluss. Es bedarf noch weiterer Recherche, um herauszufinden, ob auch ohne automatische Zeiterfassung über bereits vorhandene Android-Methoden darauf zugegriffen werden kann.

Home

Die Klasse FilterableRecyclerAdapter, die wir erstellt haben, sorgt dafür, dass die Items in einer RecyclerView angezeigt werden können, aber auch dafür, dass man nach Artikeltiteln suchen und nach Kategorien filtern kann. Dabei gab es häufig ein Timing-Problem. Dem FilterableRecyclerAdapter wird eine Liste mit den gesamten Items der Community übergeben. Der Adapter selbst ruft aber Methoden der Items auf, obwohl die Liste noch nicht fertig initialisiert wurde (NullPointer-Exception). Die versuchten Lösungswege mit Threats haben nicht funktioniert, aber wir konnten den Fehler mit einer (unschönen) try-catch Verschachtelung inkl. wait-Methode auf minimale Wahrscheinlichkeit dezimieren. Es ist nicht klar, ob dieser wieder (häufiger) auftreten wird, wenn sehr viele Artikel in der Community sind und geladen werden müssen.

6. Lessons learned

Trotz des hohen Planungsaufwandes, den wir im Vorfeld unternommen hatten, wurden wir während der Programmierung der Anwendung doch gelegentlich überrascht, wie zeitintensiv manche Module wurden. Oft mussten wir lange recherchieren, um uns den Code zu erarbeiten. Hier stellten wir auch recht schnell fest, dass die Dokumentationen der Libraries im Regelfall die beste Anlaufstelle bei Fragen waren. Doch ohne erklärende Worte zu Klassen oder deren Methoden waren Fehler schwer nachzuvollziehen und wir mussten erheblich mehr Zeit für das Debuggen aufwenden. Die Folge waren vereinzelte zeitintensive Entwicklungssprints, um die Module rechtzeitig abzuschließen. Daher hätten wir im Vorfeld mehr Zeit für Recherche und das Debuggen einplanen müssen.

Vor allem eine ausgedehntere Recherche zur Planung der Entwicklung wäre hilfreich gewesen: Hätten wir direkt Firestore statt die Firedatabase verwendet, hätten wir viel Zeit einsparen können. Außerdem hätten wir dann auch einige Module durch mehr Fragmente und weniger Activities umgesetzt.

Die Lernkurve des Programmierens war erheblich. Durch zunehmende Erfahrung mit Code und Software wurden wir wesentlich schneller bei der Umsetzung der Arbeitspakete. Zusätzlich konnten wir so auch im Laufe des Projektes besser abschätzen, wie viel Workload bestimmte Bereiche (noch) benötigen würden.

Im Laufe der Entwicklung kamen einige Funktionen dazu, die wir umsetzen wollten, die aber nicht unbedingt notwendig für die App waren. Deshalb haben wir uns mehrmals im Laufe des Projektes beraten, welche Funktionen der Nutzer wirklich benötigt und geplante Funktionen wieder verworfen. Das differenzierte Betrachten zwischendurch, wo wir sind und was

eigentlich unser Ziel war, hat dazu beigetragen, dass unsere Anwendung auch ohne Handbuch intuitiv zu bedienen ist.

7. Ausblick und mögliche Erweiterungen

Zum derzeitigen Entwicklungsstand ist die App zugeschnitten auf eine Gemeinschaft bzw. Nachbarschaftsgruppe, in der sich die Leute bereits kennen und den Communitynamen und ggf. den Zugangscode austauschen können. Es ist zwar möglich, einer Community beizutreten, die keinen Zugangscode verlangt, man kann aber nicht nach Communities in seiner Umgebung suchen, sondern muss sich anhand von Communitynamen und -beschreibung orientieren. Eine mögliche Verbesserung wäre daher, den Communities mindestens eine Postleitzahl bei der Erstellung zu geben oder ein Suchsystem zu entwickeln, das nach Communities in der Nähe der eigenen Adresse sucht. Um das Einladen der Nachbarn in eine erstelle Community zu vereinfachen, wäre es nützlich, ein vorgefertigten Aushang als PDF zum Download bereitzustellen oder einen Einladungslink zu generieren, den man digital weiterschicken kann.

Zurzeit ist der Wechsel zwischen Home-, Profil-, und Chats-View noch nicht schön visualisiert, da die gesamte Activity inklusive Bottom-Navigation neu geladen wird. Diese Activities könnte man noch in Fragments umschreiben.

Außerdem kann man beim Chat überlegen, ob es für die Nutzer doch sinnvoller wäre, nur einen Chat mit einem anderen Nutzer zu haben, anstatt einen je Artikel. Zudem könnte man auch einzelne Nachrichten löschbar machen oder dem User die Möglichkeit geben, andere zu blockieren. Damit der Chat etwas übersichtlicher bzw. schöner wird, wäre eine Nachrichtenunterteilung nach Tagen bzw. Datum sinnvoll, so müsste man nur noch die Uhrzeit unter jeder Message anzeigen (wie bei WhatsApp).

Ein weiterer wichtiger Punkt wäre das Aufrufen von den eigenen Items und Community-Details in der Profil-View inklusive möglicher Bearbeitungsfunktion. Bisher haben wir auch noch keine Löschfunktion für eine Community eingefügt. Um diese Funktion bereitzustellen, müsste ein Adminsystem eingeführt werden, oder nur der Ersteller dürfte das Recht bekommen, die Community wieder zu löschen.

Im weiteren Verlauf werden wir die App mit der Hausgemeinschaft unserer Betreuerin Sabine Schumann testen und bei der Nutzung begleiten. So können wir auch noch nicht bekannte Bugs beseitigen und die Anwendung, falls nötig, noch weiter auf die Zielgruppe zuschneiden.

Quellenverzeichnis

Firebase: https://firebase.google.com/docs

Firebase - Firestore Doku: https://firebase.google.com/docs/firestore

Firebase - Authentication Doku: https://firebase.google.com/docs/auth

Firebase - Storage Doku: https://firebase.google.com/docs/storage

Android Studio - Android Entwicklungsumgebung: https://developer.android.com/studio

FancyToast-Android Library: https://github.com/Shashank02051997/FancyToast-Android

Picasso - Android Library: https://square.github.io/picasso/

Android PDF Viewer - Library: https://github.com/barteksc/AndroidPdfViewer

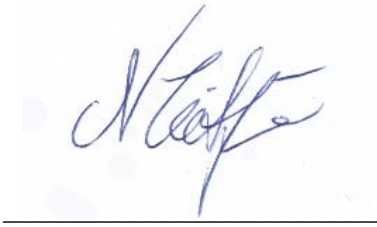
Abbildungen

Alle Abbildungen, Tabellen und Grafiken wurden selbst erzeugt und verletzen keine Rechte Dritter.

Eigenständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

Dominik Stecher, Hamburg 31.07.2019



Nina Tietje, Hamburg 31.07.2019