
**Entwicklung einer mobilen Applikation zur Terminbestimmung und -verwaltung von
Veranstaltungen zwischen Mitarbeitern im Unternehmen**

Auszubildender Fachinformatiker für Anwendungsentwicklung
Toni Kozarev
Projektzeitraum vom 20.03.2018 bis 20.04.2018



Auszubildender

Name: Toni Kozarev
Prüflingsnummer: 40059
Geburtsdatum: 13.05.1996
Anschrift: Düsternortstraße 110
27755 Delmenhorst

Ausbildungsbetrieb

Dataport Anstalt öffentlichen Rechts

Niederlassung: Bremen
Anschrift: Utbremer Str. 90
28217 Bremen

Inhaltsverzeichnis

1	Einführung	1
1.1	Unternehmensprofil	1
1.2	Themenschwerpunkte der Ausbildung	1
1.3	Projektbeschreibung	1
2	Projektvorbereitung	2
2.1	Ist-Analyse	2
2.2	Soll-Konzept	2
2.3	Kosten-Nutzen-Analyse	2
2.4	Testplanung	3
3	Projektdurchführung	4
3.1	Auswahl der Software	4
3.1.1	Auswahl der Entwicklungsumgebung	4
3.1.2	Auswahl der Frameworks	4
3.1.3	Auswahl der UI-Frameworks	5
3.1.4	Auswahl der Tests Frameworks	5
3.2	Entwicklung des UI Designs der Mobile-App	6
3.3	Implementierung der Klassen	6
3.3.1	Implementierung der Login/Register	7
3.3.2	Implementierung der Mainpage/Profile	7
3.3.3	Implementierung der anderen Klassen	9
3.3.4	Implementierung anderer Komponenten	10
3.4	Entwurf der UML-Klassendiagramme	10
3.5	Entwickeln der automatisierten Tests	10
3.6	Deployen der Mobile-App	10
4	Projektabschluss	11
4.1	Testdurchführung	11
4.2	Soll-Ist-Vergleich	11
4.3	Fazit	11
	Anhang	12
A	Glossar	12
B	Literaturverzeichnis	B-1
C	Code-Anhänge	C-1
D	Abbildungen	D-1

Tabellenverzeichnis

1	Tesplanung der Integrationstests	3
2	Soll-Ist-Vergleich	11

Liste von Code-Anhängen

1	JSON-Datei	C-1
2	Englische Übersetzung der App	C-2
3	Deutsche Übersetzung der App	C-2
4	StartappTests	C-3
5	RegisterTests	C-4
6	LoginTests	C-5
7	MainpageTests	C-7
8	InviteMembersTests	C-8
9	EventsTests	C-9
10	SearchTests	C-11
11	Start der App	C-12
12	AnmeldungDesign	C-13
13	MainpageDesign	C-14
14	ProfileContentDesign	C-16
15	ProfileDesign	C-16
16	CreateEventDesign	C-20
17	Events	C-22
18	EditEvent	C-23
19	ListView	C-30
	./code/java/Startapp.java	C-31
20	Anmeldung	C-32
21	Registrierung	C-34
22	Hauptseite	C-37
23	Profileseite	C-38
24	Teilnehmer einladen	C-45
25	Event erstellen	C-48
26	Events	C-51
27	Event bearbeiten	C-53
28	Suche	C-60
29	Suche Benutzerinformationen	C-62

Abbildungsverzeichnis

1	Firebase Authentifikation	5
2	Screenshots: Hauptmenü, Profileseite, Events	6
3	Screenshots: Splashscreen, Startseite, Registrierung	D-1
4	Screenshots: Anmeldung, Hochladen vom Bild, Profileseite mit FAB	D-1
5	Screenshots: Event erstellen, Frist ansetzen, Teilnehmer einladen	D-2
6	Screenshots: Event als Admin/Teilnehmer bearbeiten, Event verlassen	D-2
7	Screenshots: Event löschen, Mitarbeiter suchen, Visitenkarte	D-3
8	Screenshots: Hintergrundbild, Mockup Beispiel, Mockup Realisierung	D-3

9	Screenshots: Firebase TestLab - Robo-Test, alle erzeugte Tests	D-4
10	Alle UML-Klassen	D-5
11	Alle Klassen, Funktionen und Variablen	D-6

1 Einführung

Diese Projektdokumentation beschreibt die Planung, Implementierung und das Testen einer Mobile-App zur Terminabstimmung und -verwaltung von Veranstaltungen zwischen Mitarbeitern im Unternehmen. Die Projektdokumentation berücksichtigt neben der Auswahl passender Frameworks dabei wichtige Aspekte, wie eine Kosten-Nutzen-Analyse und den Datenschutz.

1.1 Unternehmensprofil

Das Unternehmen Dataport stellt für die öffentliche Verwaltung IT bereit und bietet den Behörden ein umfassendes Angebot von Dienstleistungen wie IT-Beschaffung, Fortbildungen und Schulungen [1]. Neben den oben aufgelisteten Dienstleistungen werden Datensicherheitskonzepte, sowie E-Government-Lösungen und Anwendungen für Verwaltungsaufgaben angeboten. Dataport ist eine Anstalt des öffentlichen Rechts.

Der Betrieb wurde am 1. Januar 2004 durch den Zusammenschluss der Datenzentrale Schleswig-Holstein mit dem Landesamt für Informationstechnik und der Abteilung für Informations- und Kommunikationstechnik des Senatsamtes für Bezirksangelegenheiten gegründet und hat seinen Sitz in Altenholz. Neben den Sitz in Altenholz hat unser Betrieb Niederlassungen in Hamburg, Bremen, Rostock, Lünenburg, Halle und Magdeburg.

Die Aufgabe unseres Unternehmens ist die Versorgung von Kommunikation- und Informationstechnik für die Trägerländer Hamburg, Bremen, Niedersachsen, Mecklenburg-Vorpommern, Schleswig-Holstein und Sachsen-Anhalt als Full Service Provider. Dataport hat momentan über 2.700 Mitarbeiterinnen und Mitarbeiter und erzielte 2017 einen Jahresumsatz von 547 Millionen Euro [2].

1.2 Themenschwerpunkte der Ausbildung

Die Schwerpunkte während der Ausbildung zum Fachinformatiker für Anwendungsentwicklung bei Dataport lagen in dem Testen von Software mit C# und in der Entwicklung von Programmen mit Java. Daneben waren weitere Ausbildungsinhalte wie Virtualisierung, Netzwerktechnik bzw. Netzwerkinfrastrukturen, Accounting und Linux vorhanden, die am Standort in Bremen vermittelt wurden.

1.3 Projektbeschreibung

Es soll ein Prototyp einer Mobile-App für die Vereinbarung und Bearbeitung eines betrieblichen Termins mit verschiedenen Personen entwickelt werden. Diese soll auf dem Android-Betriebssystem [3] lauffähig sein. Ein Event soll von jedem Mitarbeiter, der vom Ersteller eingeladen ist, bearbeitet und geändert werden. Jeder Mitarbeiter hat beschränkte Optionen, während ein Administrator sowohl alle Informationen ändern kann, als auch das Event komplett löschen oder eine neue Frist ansetzen kann. Die Android-App soll die Möglichkeit haben, dass mehrere Benutzer an verschiedenen Veranstaltungen teilnehmen und interaktiv neue Ideen einbringen können. Es ist auch möglich bei einem Event zu dem man eingeladen wurde, dieses abzulehnen. Aus dem Projekt soll hervorgehen, ob eine Fortführung des Projektes zu einem späteren Zeitpunkt sinnvoll und wirtschaftlich ist.

2 Projektvorbereitung

In diesem Kapitel wird der aktuelle Zustand des Projekts beschrieben, sowie der Soll-Zustand, der nach dem Projekt erreicht werden soll.

2.1 Ist-Analyse

Momentan verfügt die Bearbeitung eines betrieblichen Termins von mehreren Mitarbeitern bei Dataport über keine technische Umsetzung, wie bspw. eine Mobile-App. Es können derzeit Termine in MS-Outlook erstellt werden, diese können aber nur vom Ersteller geändert werden. Ein großer Teil der Mitarbeiter nutzt für die Terminabstimmung einer Veranstaltung die Webseite Doodle [4]. Die dort erstellten Events können nur vom Administrator geändert werden. Alle Personen, die den Link für das Event haben, können nur für die gegebenen Daten abstimmen, was die Vermittlung von interaktiven Ideen negativ beeinflusst.

2.2 Soll-Konzept

Zweck dieses Projekts soll eine lauffähige Mobile-App sein, die auf dem Android-Betriebssystem verwendet werden kann. Dazu soll das Android SDK verwendet werden. Diese Mobile-App soll darauf abzielen, eine Art Event Manager darzustellen. Da die Informationen und Event Details dynamisch konfigurierbar sein sollen, soll die Mobile-App über eine Datenbank, die Informationen speichert und zusätzlich eine Dateiablage für Bilder haben. Damit alle Benutzer miteinander kommunizieren können, wird die App das Mobilfunknetz nutzen, um Daten für die Events und unterschiedliche Konto Informationen auszutauschen. Dazu sollen bestimmte Frameworks ausgewählt und der Datenversand bzw. Datenempfang sichergestellt werden. Es sollen neben der Durchführung der eigentlichen Mobile-App auch Klassentests entwickelt und implementiert werden.

Darüber hinaus soll es möglich sein, dass der Zustand der App zwischengespeichert werden kann, sofern keine Internetverbindung besteht. Bei erneuter Verbindung sollen diese Daten gespeichert werden können.

2.3 Kosten-Nutzen-Analyse

Für dieses Projekt werden insgesamt 70 Arbeitsstunden geplant. In diesem Zeitraum muss das ganze Projekt entworfen, implementiert und getestet werden. Am Ende soll ausführlich dokumentiert werden. Es werden durch die Entwicklung der Mobile-App weder Anschaffungskosten neben dem Arbeitslohn des Anwendungsentwicklers noch laufende Kosten anfallen. Die Rechnung entspricht $70\text{Euro} * 70\text{Std.} = 4900\text{Euro}$. Die Entwicklungskosten für dieses Projekt belaufen sich auf 4900€. Die Events sind kein Teil des Projektes und werden von Dritten erstellt, deswegen fließen sie nicht mit in die Projektkosten ein.

Die Installation und Einrichten von der App sollte nicht mehr als 5 Minuten dauern. Danach sollte jeder Mitarbeiter ein Konto erstellen und seine Informationen ausfüllen. Dieser Ablauf wird kaum 5 Minuten dauern. Um ein Event zu erstellen, braucht jede Person 5-10 Minuten abhängig von der Information, die auszufüllen ist. Die Zeit-Ersparnis bei der Benutzung der Mobile-App ist ungenau zu berechnen, da jeder Mitarbeiter unterschiedlich viele Informationen einträgt bzw. zu einem späteren Zeitpunkt Informationen bearbeitet. Die Vorteile solch einer App gegenüber MS-Outlook oder Doodle sind, dass jeder Teilnehmer, der einen Zugang zu der Veranstaltung hat, kann die Informationen selber bearbeiten und muss nicht immer den Administrator kontaktieren.

2.4 Testplanung

Die Testfälle für diese Android-App sind so erstellt, dass es für jede Java Klasse eine Testsuite gibt. Alle Tests sind automatisiert und werden mithilfe des Frameworks Espresso gemacht. Es gibt Tests, die zum Beispiel die Aktionen: ein Benutzer registrieren oder anmelden, ein Event suchen, ändern oder löschen, Benutzer für ein Event einladen, die Floating Action Buttons (FAB Menu) testen. Dabei werden ebenfalls auch triviale Tests durchgeführt, die Tests sollen überprüfen, ob jede Aktion, die an einen Button gebunden ist, erfolgreich ausgeführt wird. Darüber hinaus soll überprüft werden, ob die zusätzlich implementierte Navigation korrekt funktioniert. Es werden noch verschiedene UI Elemente getestet, u.a.:

- Layout
- Button
- Edit Text
- List View
- Search View

Weitere Informationen bzgl. der Tests, siehe Unterabschnitt 3.5.

#	Testszenarios	einzelne Testfälle
1	Start der App	Funktionieren die Buttons korrekt?
2	Anmelden	Ist eine Anmeldung erfolgreich oder nicht? Ist die Email/Passwort gültig?
3	Registrieren	Ist die Registrierung erfolgreich oder wurde die Email schonmal benutzt?
4	Menü	Funktionieren des Abmeldungsbuttons und das Menü fehlerfrei?
5	Profil	Funktionieren die Texteingaben und speichern sie die neuen Informationen?
6	Benutzer einladen	Kann jeder Benutzer für einen Termin eingeladen werden?
7	Event erstellen	Kann ein Mitarbeiter ein Event erstellen?
8	Events	Ist das Suchen eines Termins möglich?
9	Event ändern	Event bearbeiten/verlassen vom Benutzer oder löschen vom Administrator
10	Benutzer suchen	Nach einem Mitarbeiter suchen und das erste Ergebnis öffnen
11	Visitenkarte öffnen	Funktioniert die Navigation?

Tabelle 1: Tesplanung der Integrationstests

3 Projektdurchführung

Im folgenden Kapitel werden alle Schritte bei der Implementierung der Android-App beschrieben und welche unerwarteten Hindernisse bzw. Probleme bei der Realisierung der geforderten Anforderungen auftraten.

3.1 Auswahl der Software

Die Auswahl der richtigen Software und Frameworks ist ein wichtiger Teil des Projektes. Die Entwicklungsumgebung Android Studio [5] ist eine sehr beliebte Umgebung für die Erzeugung von Android Applikationen und wird meist empfohlen. Außerdem ist die Dokumentation des Firebase [6] Frameworks, die auch von Google stammt, sehr verständlich und ausführlich beschrieben, sowie öffentlich zugänglich und kommt mit vielen Beispielen.

3.1.1 Auswahl der Entwicklungsumgebung

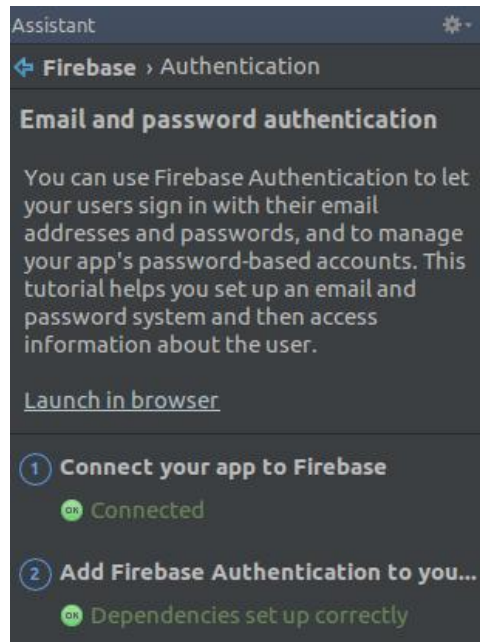
Für die Implementierung der Mobile-App wurde sich für die Entwicklungsumgebung Android Studio entschieden. Diese Umgebung wird empfohlen für den Einsatz der Android SDK und bringt bereits Gradle [7] mit, das für die Anwendung der Mobile-App benötigt wird.

3.1.2 Auswahl der Frameworks

Am Anfang des Projektes wurde sich für bestimmte Frameworks entschieden, die für die Durchführung des Projektes fundamental sind. Dazu zählt Firebase als ein integriertes Framework im Android Studio, siehe Abbildung 1, das für die Authentifizierung der registrierten/angemeldeten Benutzer, Aufbewahrung von Bildern und der Datenbank Anbindung die für restliche Informationen benötigt wird. Damit die ausgewählten Frameworks benutzt werden können, müssen sie in der Gradle-Konfiguration eingerichtet werden. Dies ist nur möglich, wenn die **build.gradle** Konfigurationsdatei mit der unter stehenden Zeilen erweitert werden. Nach der Implementierung einer Dependency haben alle Klassen und Methoden Zugriff auf die enthaltenen Funktionalitäten in den Frameworks. Momentan ist die Version 11.0.4 aktuell und sie sollte für alle Firebase Frameworks benutzt werden.

```
implementation 'com.google.android.gms:play-services-auth:11.0.4'  
implementation 'com.google.firebase:firebase-auth:11.0.4'  
implementation 'com.google.firebase:firebase-storage:11.0.4'  
implementation 'com.google.firebase:firebase-database:11.0.4'  
implementation 'com.google.firebase:firebase-core:11.0.4'
```


Abbildung 1: Firebase Authentifikation



3.1.3 Auswahl der UI-Frameworks

Für das UI Design werden andere Bibliotheks-Abhängigkeiten (Dependency – pl. Dependencies) wie CardView, Glide und ConstraintLayout genutzt. Das erste wurde für das Hauptmenü angewendet. Das Menü besteht aus 6 Bildschaltflächen, siehe Abbildung 4, aber nur 4 von den haben eine Funktionalität. Die anderen 2 sind momentan ohne Funktion, aber die werden zu einem späteren Zeitpunkt weiterentwickelt. Glide wurde nur für das Einspielen von Profilbildern verwendet, die in der Firebase Lagerung gespeichert wurden. Das ConstraintLayout Dependency wurde für das Design die Aktivitäten genutzt.

```
implementation 'com.android.support:cardview-v7:26.0.1'  
implementation 'com.github.bumptech.glide:glide:4.3.1'  
annotationProcessor 'com.github.bumptech.glide:compiler:4.3.1'  
implementation 'com.android.support.constraint:constraint-layout:1.1.0-beta4'
```

3.1.4 Auswahl der Tests Frameworks

Außerdem wird Firebase TestLab [8] für einen Robo-Test benutzt. Dieser Test überprüft, ob eine bestimmte App auf einem bestimmten Smartphonemodell fehlerfrei funktioniert. Siehe Abbildung 9. Neben den allen bisher genannten Frameworks wurde ein weiteres genutzt, das Framework Espresso [9]. Dieses Framework wurde für die Erstellung der automatisierten Tests benutzt. Diese UI-Tests liefen auf einem Smartphone oder Emulator und überprüfen, ob bei einer Änderung am User Interface das gleiche Ergebnis vorkommt. Diese Tests werden auch Instrumented Tests genannt.

```
androidTestImplementation 'com.android.support.test:runner:1.0.1'  
androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
```

Die verwendeten Frameworks sind Open Source Software und es fallen keine Kosten hierfür an.

3.2 Entwicklung des UI Designs der Mobile-App

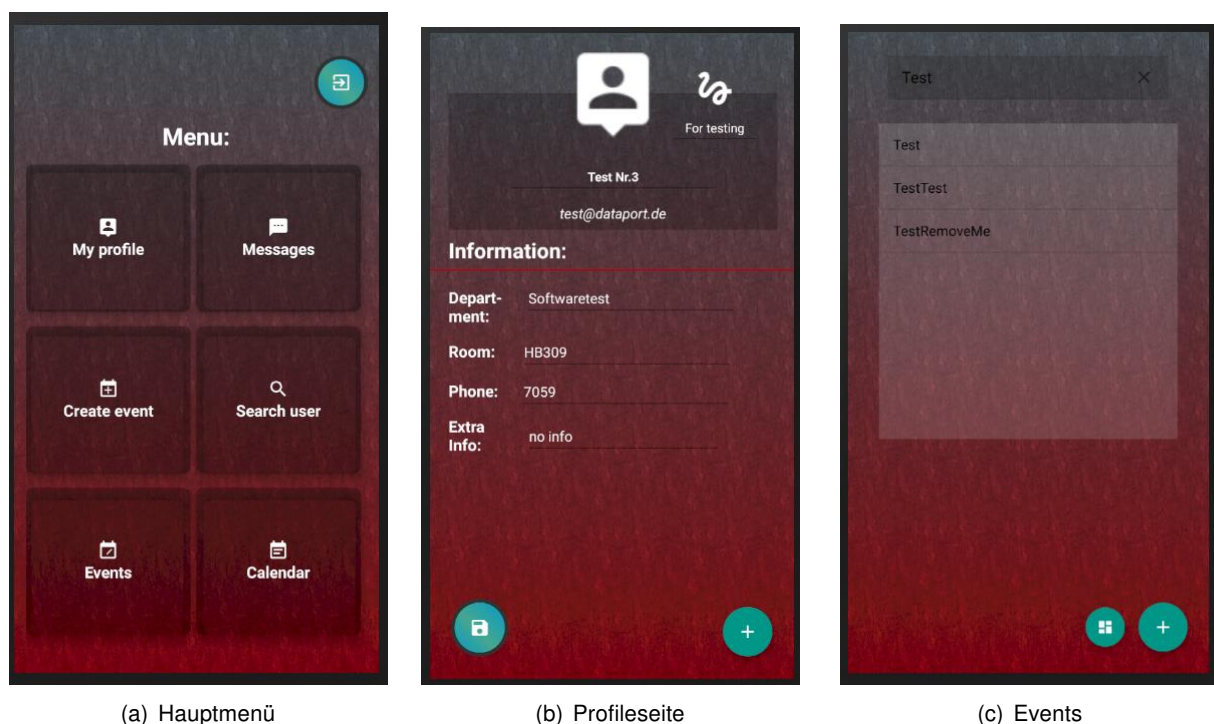
Als Erstes wird ein sogenanntes Mockup der App erstellt. Ein Mock-Up bedeutet einen Wegwerfprototyp der Benutzerschnittstelle einer zu erstellenden Software. Die Abbildung 8 stellt ein Beispiel für ein Mockup dar, die für das Projekt angefertigt wurde. Auf dem Screenshot kann man sehen, wie die Positionierung der einzelnen Elemente geplant war und berücksichtigt wird, dass die Elemente sich nicht überschneiden. Die Schriftgröße der Texte und die Größe der Menü Buttons müssen auch entsprechend gewählt werden, um das Lesen und das Navigieren in der Mobile-App zu erleichtern. Die Implementierung dieses Mockups wird in der Abbildung 8 repräsentiert. Für das Design der App wurden viele freie Icons von Google Material [10] benutzt. Alle Elemente werden hauptsächlich durch die XML-Dateien generiert, aber sie können auch in dem Java-Code geändert werden.

Um die App universell und speziell für unser Unternehmen zu machen und keine populären Layouts zu benutzen, wurde das ganze Projekt in einem Vollbildansicht-Thema gemacht. Dazu wurde ein Navigationsmenü (FAB Buttons) mit einer Animation hinzugefügt, um die App nutzerfreundlicher zu machen. Es werden meistens Farben benutzt, die im Unternehmens Logo, auf der Webseite und in den Dokumenten von Dataport verwendet werden.

3.3 Implementierung der Klassen

In diesem Kapitel wird die ganze Implementierung des Projektes beschrieben. Beim Starten der App wird ein Splashscreen aktiviert [11]. Nach dem Laden der Applikation sollte eine Anmeldungs- maske erzeugt werden. Um Mitarbeiter anzumelden, sollten sie vorher registriert werden. Dazu wird es beim Starten von der Applikation eine Aktivität geben, wo beide Optionen auftauchen. Wenn ein Konto erstellt wurde, wird die Person zu einer neuen Aktivität weitergeleitet, wo sich das Hauptmenü befindet.

Abbildung 2: Screenshots: Hauptmenü, Profileseite, Events



Jeder angemeldete Nutzer hat die Möglichkeit seine Profil-Daten und Bilder zu ändern. Ein Nutzer kann auch ein Event erstellen oder bearbeiten. Dazu werden noch zwei neue Aktivitäten realisiert. Bei der Erstellung von einer Veranstaltung soll der Nutzer zuerst die Teilnehmer auswählen, die zu diesem Event eingeladen werden sollen. Danach gibt der Nutzer die Informationen für das Event ein. Alle Termine, die von dem eingeloggtten Konto zu sehen sind, werden auf die zweite Aktivität - **Events** als eine Liste auftauchen. Dort können die Mitarbeiter ein Event nach dem Name suchen und öffnen. Wenn der Ersteller der Veranstaltung die angemeldete Person ist, dann hat diese Person die Berechtigung Informationen zu ändern, eine neue Frist anzusetzen oder sogar das ganze Event zu löschen. Ein eingeladener Nutzer kann die Texteingaben bearbeiten oder das Event verlassen. Es wurde noch eine vierte Aktivität implementiert. Dort werden alle registrierten Konten aufgelistet. Jede Person kann ein Konto anhand einer Email-Adresse suchen. Beim Öffnen von einer Visitenkarte hat der Benutzer zugriff zu den Daten seiner Kollegen. Am Ende sollte ein Abmeldungsbutton hinzugefügt werden.

3.3.1 Implementierung der Login/Register

Die Implementierung der Anmeldungs- und Registrierungsaktivitäten fand mit dem Einrichtung des Firebase Frameworks Abbildung 1 statt. Für die Anmeldung und Registrierung von Mitarbeitern wurde Firebase Authentifikation benutzt. Damit die App einen Zugriff mit dieser Anmeldung realisieren kann, sollte eine Instanz zu einem **FirebaseAuth**-Objekt umgesetzt werden, wie unten abgebildet:

```
24 private FirebaseAuth mAuth;  
25 ...  
26 @Override  
27 protected void onCreate(Bundle savedInstanceState) {  
28     mAuth = FirebaseAuth.getInstance();  
29     ...  
30 }
```

Für die Registrierung sollte jede Person ihre Email und zweimal das gleiche Passwort eingeben. Wenn das Passwort übereinstimmt und die Email valide ist, wird auf dem **mAuth**-Objekt die Methode - **createUserWithEmailAndPassword** aufgerufen. Wenn diese Methode erfolgreich ausgeführt wurde, wurde ein neues Konto mit der eingegebenen Email-Adresse und Passwort erstellt. Jeder registrierte Nutzer erhält eine User-ID, die auch in der Firebase Authentifikation gespeichert wird. Firebase Authentifikation erlaubt keine doppelten Anmeldungen mit der gleichen Email-Adresse, deswegen ist jede Email-Adresse genauso wie die UID universell und eindeutig.

Für die Anmeldung sollte die gleiche Prozedur durchgeführt werden. Hier wurde die Methode **signInWithEmailAndPassword** auf dem **mAuth**-Objekt aufgerufen. Bei einer erfolgreichen Anmeldung sollte eine neue Aktivität geöffnet werden.

3.3.2 Implementierung der Mainpage/Profile

Nach der Implementierung des Anmeldebildschirms sollte das Hauptmenü implementiert werden. Für dieses wurde eine CardView-Ansicht erstellt [12]. Das Menü wurde auf 6 Felder aufgeteilt und jedes Feld erhält eine Abbildung und einen Text.

```
58 <GridLayout ... >  
59     <!-- Row 1, Column 1 -->  
60     <android.support.v7.widget.CardView ... >  
61         <LinearLayout ... >  
62             <ImageView ... />
```

```
63         <TextView ... />
64     </LinearLayout>
65 </android.support.v7.widget.CardView>
66
67 <!-- Row 1, Column 2 -->
68 <android.support.v7.widget.CardView ... >
69     ...
70 </android.support.v7.widget.CardView>
71 </GridLayout>
```

In Java würde das Menü so aussehen:

```
20 GridLayout mainGrid;
21 ...
22 @Override
23 protected void onCreate(Bundle savedInstanceState) {
24     mainGrid = findViewById(R.id.mainGrid);
25     ...
26 }
27
28 private void setSingleEvent(GridLayout mainGrid){
29     for(int i = 0; i < mainGrid.getChildCount(); i++){
30         CardView cV = (CardView)mainGrid.getChildAt(i);
31         final int var = i;
32         cV.setOnClickListener(new View.OnClickListener() {
33             @Override
34             public void onClick(View view) {
35                 if(var == 0){
36                     Intent i = new Intent(Mainpage.this, Profile.class);
37                     startActivity(i);
38                 }else if(var == 4){
39                     Intent i = new Intent (Mainpage.this, Events.class);
40                     startActivity(i);
41                 }else{
42                     ... //die andere Teile der Menü
43                 }
44             }
45         });
46     }
47 }
```

Für die Erstellung der Profilseite wurde eine Datenbank für die Speicherung der Informationen genutzt. Die Datenbank für diese Applikation ist wieder von Firebase und es werden die Daten in einer JSON-Datei gespeichert. Um eine Verbindung mit der Datenbank herzustellen, muss ein **mDatabase**-Objekt erstellt werden. Das Objekt ist vom Typ **FirebaseDatabase** und soll ähnlich wie bei der Firebase Authentifikation eine Instanz auf dem Objekt erzeugen. Für die Verwaltung von den Profilbildern wurden noch zwei Frameworks - Firebase Storage und Glide benutzt.

Um ein Bild hochzuladen, wurde die Funktion **uploadImage()** erstellt, die eine Instanz auf ein **Firebase Storage**-Objekt erzeugt. Danach wurde das Bild mit der Methode **putFile** hochgeladen.

Um das Bild zu speichern, sollte die Methode **setPhotoUri** auf dem Objekt vom Typ **UserProfileChangeRequest.Builder** genutzt werden. Danach sollten die neuen Änderungen mit der Methode **updateProfile** aktualisiert werden. Auf diese Weise soll das Bild erfolgreich gespeichert werden.

```
291 String imageUrl;
292 ...
293 UserProfileChangeRequest profile = new UserProfileChangeRequest.Builder()
```

```
294         .setPhotoUri(Uri.parse(imageUrl))
295         .build();
296 user.updateProfile(profile);
297 ...
```

Um das Bild nach einem erneuten Öffnen der Profilseite zu laden, wurde Glide wie folgt benutzt:

```
277 ImageView profileInfoPhoto;
278 ...
279 private void loadUserPhoto() {
280     FirebaseUser user = mAuth.getCurrentUser();
281     if (user != null) {
282         if (user.getPhotoUrl() != null) {
283             Glide.with(this)
284                 .load(user.getPhotoUrl().toString())
285                 .into(profileInfoPhoto);
286         }
287     }
288 }
```

Bei der Profilseite gibt es noch einige Texteingaben, wo jeder Benutzer mehr Informationen über sich selber eingeben kann. Um die Eingaben in einer JSON-Datenbank zu speichern, sollten sie in verschiedenen Variablen gespeichert werden und danach die Methode **child()** auf der **DatabaseReference** benutzen, um die richtige Stelle für den Wert in der Datenbank zu finden.

```
50 FirebaseDatabase mDatabase = FirebaseDatabase.getInstance();
51 DatabaseReference mRef = mDatabase.getReference("Benutzer/");
52 EditText profileInfoName;
53 ...
54 String name = profileInfoName.getText().toString();
55 mRef.child(userID).child("Name").setValue(name);
```

Um Informationen aus der Datenbank zu bekommen, muss die **DatabaseReference** der Methode **addValueEventListener** aufgerufen werden. Danach wurde auf die Funktion **onDataChange** mithilfe der *dataSnapshot* der richtige Text von der JSON-Datei ermittelt und die Profilseite aktualisiert.

```
235 DatabaseReference mRef = mDatabase.getReference("Benutzer/");
236 ...
237 mRef.addValueEventListener(new ValueEventListener() {
238     @Override
239     public void onDataChange(DataSnapshot dataSnapshot) {
240         String name = dataSnapshot.child("Name").getValue(String.class);
241         profileInfoName.setText(name);
242     }
243 }
```

3.3.3 Implementierung der anderen Klassen

Alle anderen Klassen wurden identisch zu der Profile-Klasse implementiert. Für jede Klasse können die Mitarbeiter Informationen von der Datenbank hinzufügen, ändern oder löschen. In der Events-/InviteMembers-/Search-Klassen wurden Listen mit allen Events/Benutzer aufgezeigt. Jeder kann ein Event suchen, wenn er Zugriff zu dieser Veranstaltung hat.

3.3.4 Implementierung anderer Komponenten

Es werden andere Komponenten wie Date Dialog und Floating Action Buttons implementiert. Die Aufgabe der FAB ist eine schnelle Navigation zu erzielen, während Date Dialog für die Auswahl der Frist eines Events zuständig ist.

3.4 Entwurf der UML-Klassendiagramme

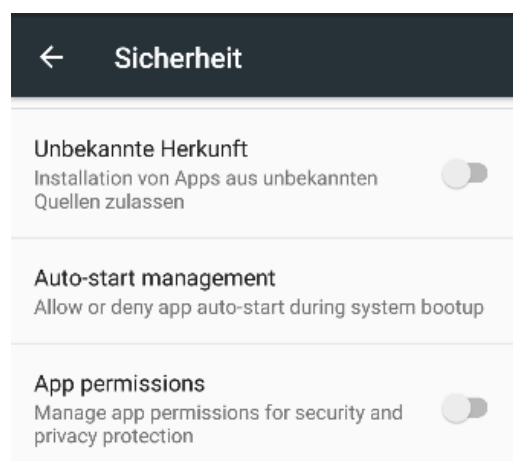
Vor der Implementierung der Klassen wurde ein UML-Klassendiagramm per Hand gezeichnet. Nach dem Entwurf aller Java Klassen mithilfe des Diagramms wurde ein Plugin im Android Studio benutzt, um alle Variablen und Funktionen jeder Klasse darzustellen. Diese werden in Abbildung 10 und Abbildung 11 gezeigt. Mithilfe eines Klassendiagramms konnte berücksichtigt werden, welche der geplanten Methoden implementiert und welche Variablen benötigt wurden.

3.5 Entwickeln der automatisierten Tests

In diesem Ausschnitt wird die Erstellung der Tests betrachtet. Anhand des fertigen Designs und der implementierten Java Klassen konnten einige Tests entworfen werden. Es handelt sich um automatisierte Tests, die nicht nur die Funktionalität der Oberfläche, sondern auch die Funktionalität der Methoden überprüfen können. Der Code-Anhang im Unterabschnitt C-9 zeigt, dass ein Test nicht nur einen Fehler auslösen kann, wenn ein Wert nicht korrekt ist, sondern auch die Richtigkeit der Funktion bei der Eingabe von korrekten Werten überprüfen kann. Einige Tests überprüfen nur das Navigationsmenü oder die Texteingaben und Buttons, während andere überprüfen, ob zum Beispiel die Anmeldung erfolgreich ist. Es prüft auch, ob ein Benutzer sich mit einer Email-Adresse registrieren kann, wenn die Email-Adresse schon benutzt wurde oder ob das Passwort die Bedingungen vom Firebase Authentifikation [13] erfüllt. Wenn ein Passwort nicht vorhanden ist oder weniger als sechs Zeichen hat, ist keine Registrierung möglich.

3.6 Deployen der Mobile-App

Um eine Android-App zu installieren und anwenden zu können, muss zuerst eine .apk-Datei erzeugt werden. Es gibt eine Release-Version und eine Debug-Version. In beiden Fällen wird Gradle als Buildsystem benutzt. Die App soll intern genutzt werden und von daher soll die Sicherheitseinstellung für unbekannte Quellen aktiviert werden. Siehe Abbildung unten.



4 Projektabschluss

4.1 Testdurchführung

Um die Tests durchzuführen muss ein Emulator installiert oder ein Android-Gerät angeschlossen sein. Alle Tests sind automatisiert und werden mithilfe des Espresso Frameworks Dependency durchgeführt. Diese sogenannten Instrumented Tests können mit dem Befehl **gradlew connectedAndroidTest** ausgeführt werden. Test-Ergebnisse werden mit dem Einsatz vom Gradle-Befehl produziert. Für die Tests wurde ein Emulator mit der Android-Version 7.0 („Nougat“) [14] verwendet. Auf der Konsole wurden alle Testszenarien und alle einzelnen Testfälle durchgeführt. Dort ist zu sehen welche und wie viele Tests erfolgreich bzw. fehlgeschlagen sind. Dazu werden die Tests nochmal auf einem Android-Handy mit der Version 6.0 („Marshmallow“) [15] durchgeführt. Auf der Abbildung 9 ist ein Teil der Tests zu sehen. Die Durchführung der Testszenarien verlief wie geplant. Alle 41 Tests sind automatisiert und waren auf Smartphone und Emulator 100% erfolgreich.

4.2 Soll-Ist-Vergleich

In der folgenden Tabelle wird der Verlauf des Projekts und die aufgewendeten Stunden mit der Planung verglichen. Der Teil der Projektplanung und der Durchführung benötigten jeweils eine Stunde mehr Zeit als gedacht, wodurch die Projektplanung von 5 auf 6 Stunden und die Implementierung der App von 47 auf 48 Stunden angestiegen sind. Diese Stunden wurden in der Dokumentations-Phase eingespart, da die Tests schnell und fehlerfrei verliefen. Entgegen der Planung im Projektantrag konnten die Funktionen Versenden von Nachrichten und Kalender Vorschau nicht umgesetzt werden, da dieses den geplanten Zeitrahmen deutlich überschritten hätte. Somit wurden die 70 Arbeitsstunden eingehalten.

Projektphase	Soll-Stunden	Ist-Stunden
Projektplanung	5	6
Projektdurchführung	47	48
Test und Dokumentation	18	16
Gesamtsumme	70	70

Tabelle 2: Soll-Ist-Vergleich

4.3 Fazit

Das Projekt „Entwicklung eines Mobile-App-Prototypen zur Terminabstimmung und -verwaltung von Veranstaltungen zwischen Mitarbeitern im Unternehmen“ konnte mit reduziertem Leistungsumfang erfolgreich abgeschlossen werden. Als Resultat ist eine Android-App entstanden, mit dessen Hilfe viele Mitarbeiter verschiedene Events bearbeiten können.

A Glossar

.apk Die Installationsdatei für eine Android-Applikation.

Android Ein Betriebssystem für Smartphones.

Android SDK Android Entwicklerwerkzeug.

Aktivität Die Ansicht einer Android-Applikation.

Dependency Bibliotheks-Abhängigkeit für ein Framework.

Deployen Die Bereitstellung einer Android-Anwendung.

Floating Action Buttons Der FAB ist ein dynamisches UI-Element, das die schnelle Navigation in der Applikation unterstützt und die Benutzerfreundlichkeit verbessert.

Full Service Provider Vollständige Übernahme von Geschäftsprozessen.

Framework Eine Art externer Bibliotheken.

Gradle Build-Management-Tool.

Integrationstests Eine Menge an Tests, die die Funktionalität einer Software als ganzes testen soll.

Java Eine höhere Programmiersprache

JSON JSON ist ein kompaktes Datenformat in einer einfach lesbaren Textform.

Linux Ein freies, unix-ähnliches Mehrbenutzer-Betriebssystem.

Mockup Ein Vorführmodell, das die Funktionen eines fertigen Produktes repräsentieren soll.

Mobile-App Eine Applikation für das Smartphone.

MS-Outlook Eine Software von Microsoft zum Verwalten von Terminen und Aufgaben, sowie zum Empfangen und Versenden von E-Mails.

Open Source Software Eine Software, deren Quelltext öffentlich zugreifbar ist und vom Dritter eingesehen und genutzt werden kann.

Splashscreen Eine Aktivität, die beim Starten einer Applikation dargestellt wird.

UID User-ID.

UML-Klassendiagramm Ein Klassendiagramm ist ein Strukturdiagramm der Unified Modeling Language (UML) zur grafischen Darstellung von Klassen und ihre Beziehungen.

Virtualisierung Abstraktion von physikalischen IT-Ressourcen in Form von Hardware, Software oder Netzwerken zu virtuellen Komponenten.

B Literaturverzeichnis

- [1] D. AöR, "Dataport lösungen a-z." <https://www.dataport.de/Seiten/L%C3%B6sungen/LC3%B6sungen-A-bis-Z.aspx>. [Online, letzter Abruf: 12-April-2018].
- [2] D. AöR, "Unternehmensportät." <https://www.dataport.de/Seiten/Unternehmen/%C3%9Cber-uns.aspx>. [Online, letzter Abruf: 12-April-2018].
- [3] Google, "Android." <https://www.android.com/>. [Online, letzter Abruf: 16-April-2018].
- [4] "Doodle." <https://doodle.com/de/>. [Online, letzter Abruf: 17-April-2018].
- [5] "Android studio." <https://developer.android.com/studio/index.html>. [Online, letzter Abruf: 11-April-2018].
- [6] Google, "Firebase documentation." <https://firebase.google.com/docs/>. [Online, letzter Abruf: 18-April-2018].
- [7] G.Inc., "Gradle." <https://gradle.org/>. [Online, letzter Abruf: 18-April-2018].
- [8] Google, "Firebase tests." <https://firebase.google.com/docs/test-lab/>. [Online, letzter Abruf: 18-April-2018].
- [9] Google, "Ui-tests mit espresso." <https://developer.android.com/training/testing/espresso/index.html>. [Online, letzter Abruf: 18-April-2018].
- [10] Google, "Material icons." <https://material.io/icons/>. [Online, letzter Abruf: 15-April-2018].
- [11] "Splashscreen." <https://www.skillnotch.com/images/spinner.gif>. [Online, letzter Abruf: 22-March-2018].
- [12] Android. <https://developer.android.com/reference/android/support/v7/widget/CardView.html>. [Online, letzter Abruf: 27-March-2018].
- [13] Google, "Firebase authentifikation." <https://firebase.google.com/docs/auth/>. [Online, letzter Abruf: 16-April-2018].
- [14] Google, "Android 7.0 nougat." https://www.android.com/intl/de_de/versions/nougat-7-0/. [Online, letzter Abruf: 19-April-2018].
- [15] Google, "Android 6.0 marshmallow." https://www.android.com/intl/de_de/versions/marshmallow-6-0/. [Online, letzter Abruf: 19-April-2018].

C Code-Anhänge

JSON-Datei (Firebase Database)

```
1 {
2   "Benutzer" : {
3     "DRhznSQY5wdrWs8qPBOY3vpuDnE2" : {
4       "Abteilung" : "Softwaretest",
5       "Additional Info" : "no info",
6       "Email" : "test@dataport.de",
7       "Leitzeichen" : "For testing",
8       "Name" : "Test Nr.3",
9       "Raum" : "HB309",
10      "Telefon" : "7059",
11      "UserID" : "DRhznSQY5wdrWs8qPBOY3vpuDnE2"
12    },
13    "KDg7NdcJCYeiSqh4MPaPDhTtA5A3" : {
14      "Abteilung" : "Java",
15      "Additional Info" : "No extra info!",
16      "Email" : "bob@gmail.com",
17      "Leitzeichen" : "RK32",
18      "Name" : "Bob Goedcke",
19      "Raum" : "HB309.2",
20      "Telefon" : "4952745806",
21      "UserID" : "KDg7NdcJCYeiSqh4MPaPDhTtA5A3"
22    }
23  },
24  "Benutzer-Events" : {
25    "-LAc5izZiXNNnUuwYDa2" : {
26      "DRhznSQY5wdrWs8qPBOY3vpuDnE2" : "Admin",
27      "KDg7NdcJCYeiSqh4MPaPDhTtA5A3" : "bob@gmail.com"
28    }
29  },
30  "Events" : {
31    "-LAc5izZiXNNnUuwYDa2" : {
32      "Creator" : "test@dataport.de",
33      "Deadline" : "26/4/2018",
34      "EventID" : "-LAc5izZiXNNnUuwYDa2",
35      "Information" : "no info",
36      "ListMembers" : [ "bob@gmail.com" ],
37      "Members" : "bob@gmail.com",
38      "Name" : "TestRemoveMe",
39      "Place" : "online",
40      "Plan" : "at 12 a.m.",
41      "Price" : "no price",
42      "Purpose" : "for testing"
43    }
44  }
45 }
```

Code-Anhang 1: JSON-Datei

Englische Übersetzung

```
1 <resources>
2   <!-- Startapp and Splashscreen -->
3   <string name="welcomeTextStartapp">Welcome to \nEvent Manager</string>
4   <string name="logoDP">Dataport</string>
5   <string name="imageBtnRegisterText">Register</string>
6   <string name="imageBtnLoginText">Login</string>
7
8   <!-- Register -->
9   <string name="registerText">Create an account</string>
10  <string name="regEmail">Email</string>
11  <string name="regPass">Password</string>
12  <string name="regConfirmPass">Confirm Password</string>
13  <string name="regBtn">Sign up</string>
14
15  <!-- Login -->
16  <string name="title_activity_login">Login</string>
17  <string name="loginText">Log in to your account</string>
18
19  <!-- Mainpage -->
20  <string name="title_activity_mainpage">Mainpage</string>
21  <string name="logoutBtn">Sign out</string>
22  <string name="mainpageText">Menu:</string>
23  <string name="cardProfile">My profile</string>
24  <string name="cardSearch">Search user</string>
25  <string name="cardCreateEvent">Create event</string>
26  <string name="cardEditEvent">Events</string>
27
28  <string name="beispiel"> usw... </string>
29 </resources>
```

Code-Anhang 2: Englische Übersetzung der App

Deutsche Übersetzung

```
1 <resources>
2   <!-- Startapp and Splashscreen -->
3   <string name="welcomeTextStartapp">Willkommen bei \nEvent Manager</string>
4   <string name="logoDP">Dataport</string>
5   <string name="imageBtnRegisterText">Registrieren</string>
6   <string name="imageBtnLoginText">Anmelden</string>
7
8   <!-- Register -->
9   <string name="registerText">Registrieren Sie sich</string>
10  <string name="regEmail">Email</string>
11  <string name="regPass">Passwort</string>
12  <string name="regConfirmPass">Passwort bestätigen</string>
13  <string name="regBtn">Register</string>
14
15  <!-- Login -->
16  <string name="title_activity_login">Login</string>
```

```
17 <string name="loginText">Melden Sie sich an</string>
18
19 <!-- Mainpage -->
20 <string name="title_activity_mainpage">Mainpage</string>
21 <string name="logoutBtn">Abmelden</string>
22 <string name="mainpageText">Menu:</string>
23 <string name="cardProfile">Mein Konto</string>
24 <string name="cardSearch">Suche</string>
25 <string name="cardCreateEvent">Event erstellen</string>
26 <string name="cardEditEvent">Events</string>
27
28 <string name="beispiel"> usw... </string>
29 </resources>
```

Code-Anhang 3: Deutsche Übersetzung der App

StartappTests

```
1 package com.example.tonyk.promac;
2
3 import android.app.Activity;
4 import android.app.Instrumentation;
5 import android.support.test.rule.ActivityTestRule;
6
7 import org.junit.After;
8 import org.junit.Before;
9 import org.junit.Rule;
10 import org.junit.Test;
11
12 import static android.support.test.InstrumentationRegistry.getInstrumentation;
13 import static android.support.test.espresso.Espresso.onView;
14 import static android.support.test.espresso.action.ViewActions.click;
15 import static android.support.test.espresso.matcher.ViewMatchers.withId;
16 import static junit.framework.Assert.assertNotNull;
17
18 public class StartappTest {
19
20     @Rule
21     public ActivityTestRule<Startapp> mStartAppTestRule = new ActivityTestRule<>(
22         ↪ Startapp.class);
23     private Startapp startAppActivity = null;
24     private Instrumentation.ActivityMonitor monitorLogin = getInstrumentation().
25         ↪ addMonitor(Login.class.getName(), null, false);
26     private Instrumentation.ActivityMonitor monitorRegister = getInstrumentation().
27         ↪ addMonitor(Register.class.getName(), null, false);
28
29     @Before
30     public void setUp() throws Exception {
31         startAppActivity = mStartAppTestRule.getActivity();
32     }
33
34     @Test
35     public void testGoToLoginBtn() {
36         assertNotNull(startAppActivity.findViewById(R.id.goToLogin));
37         onView(withId(R.id.goToLogin)).perform(click());
38         Activity loginActivity = getInstrumentation().waitForMonitorWithTimeout(
39             ↪ monitorLogin, 2500);
40         assertNotNull(loginActivity);
41         loginActivity.finish();
42     }
43 }
```

```
40     @Test
41     public void testGoToRegisterBtn() {
42         assertNotNull(startAppActivity.findViewById(R.id.goToRegistration));
43         onView(withId(R.id.goToRegistration)).perform(click());
44         Activity registerActivity = getInstrumentation().waitForMonitorWithTimeout(
45             ↪ monitorRegister, 2500);
46         assertNotNull(registerActivity);
47         registerActivity.finish();
48     }
49     @After
50     public void tearDown() throws Exception {
51         startAppActivity = null;
52     }
53 }
54 }
```

Code-Anhang 4: StartappTests

**Wichtig: Alle Testklassen nach StartappTest.java sind reduziert und haben keine Imports.
RegisterTests**

```
1 public class RegisterTest {
2
3     private FirebaseAuth auth;
4
5     @Rule
6     public ActivityTestRule<Register> mRegisterTestRule = new ActivityTestRule<>(
7         ↪ Register.class);
8     private Register registerActivity = null;
9     private Instrumentation.ActivityMonitor monitorMainpage = getInstrumentation().
10        ↪ addMonitor(Mainpage.class.getName(), null, false);
11
12     @Before
13     public void setUp() throws Exception {
14         registerActivity = mRegisterTestRule.getActivity();
15         auth = FirebaseAuth.getInstance();
16     }
17
18     @Test
19     public void registration() {
20         if(auth.getCurrentUser() == null) {
21             auth.createUserWithEmailAndPassword(onView(withId(R.id.regEmail)).
22                 ↪ perform(typeText("test2@dataport.de")).toString(), onView(withId(
23                 ↪ R.id.regPass)).perform(typeText("123456")).toString());
24             closeSoftKeyboard();
25             onView(withId(R.id.regConfirmPass)).perform(typeText("123456"));
26             closeSoftKeyboard();
27             onView(withId(R.id.regBtn)).perform(click());
28             onView(withText(R.string.registrationSuccessful)).inRoot(withDecorView(
29                 ↪ not(is(registerActivity.getWindow().getDecorView()))).check(
30                 ↪ matches(isDisplayed()));
31             Activity mainpageActivity = getInstrumentation().
32                 ↪ waitForMonitorWithTimeout(monitorMainpage, 2500);
33             assertNotNull(mainpageActivity);
34             mainpageActivity.finish();
35         }
36     }
37 }
```

```
31  @Test
32  public void emailExistsRegistrationFailed() {
33      if(auth.getCurrentUser() == null) {
34          auth.createUserWithEmailAndPassword(onView(withId(R.id.regEmail)).
35              ↪ perform(typeText("test@dataport.de")).toString(), onView(withId(R
36              ↪ .id.regPass)).perform(typeText("123456")).toString());
37          closeSoftKeyboard();
38          onView(withId(R.id.regConfirmPass)).perform(typeText("123456"));
39          closeSoftKeyboard();
40          onView(withId(R.id.regBtn)).perform(click());
41          onView(withText(R.string.emailExistsRegistrationFailed)).inRoot(
42              ↪ withDecorView(not(is(registerActivity.getWindow().getDecorView())
43              ↪ ))) .check(matches(isDisplayed()));
44      }
45  }
46
47  @After
48  public void tearDown() throws Exception {
49      registerActivity = null;
50      if(auth != null) {
51          auth.signOut();
52          auth = null;
53      }
54  }
```

Code-Anhang 5: RegisterTests

LoginTest

```
1  public class LoginTest {
2
3      private FirebaseAuth auth;
4
5      @Rule
6      public ActivityTestRule<Login> mLoginTestRule = new ActivityTestRule<>(Login.
7          ↪ class);
8      private Login loginActivity = null;
9      private Instrumentation.ActivityMonitor monitorMainpage = getInstrumentation().
10         ↪ addMonitor(Mainpage.class.getName(), null, false);
11
12     @Before
13     public void setUp() throws Exception {
14         loginActivity = mLoginTestRule.getActivity();
15         auth = FirebaseAuth.getInstance();
16     }
17
18     @Test
19     public void login() {
20         if(auth.getCurrentUser() == null) {
21             auth.signInWithEmailAndPassword(onView(withId(R.id.loginEmail)).perform(
22                 ↪ typeText("test@dataport.de")).toString(), Espresso.onView(withId(
23                 ↪ R.id.loginPass)).perform(typeText("123456")).toString());
24             closeSoftKeyboard();
25             onView(withId(R.id.loginBtn)).perform(click());
26             Activity mainpageActivity = getInstrumentation().
27                 ↪ waitForMonitorWithTimeout(monitorMainpage, 2500);
28             assertNotNull(mainpageActivity);
29         }
30     }
31 }
```

```

24     }
25 }
26
27 @Test
28 public void loginFailed() {
29     if(auth.getCurrentUser() == null) {
30         auth.signInWithEmailAndPassword(onView(withId(R.id.loginEmail)).perform(
31             ↪ typeText("test@dataport.de").toString(), Espresso.onView(withId(
32             ↪ R.id.loginPass)).perform(typeText("654321").toString()));
31         closeSoftKeyboard();
32         onView(withId(R.id.loginBtn)).perform(click());
33         onView(withText(R.string.loginFailed)).inRoot(withDecorView(not(is(
34             ↪ loginActivity.getWindow().getDecorView())))).check(matches(
35             ↪ isDisplayed()));
34     }
35 }
36
37 @Test
38 public void passwordIsEmpty() {
39     if(auth.getCurrentUser() == null) {
40         auth.signInWithEmailAndPassword(onView(withId(R.id.loginEmail)).perform(
41             ↪ typeText("test@dataport.de").toString(), Espresso.onView(withId(
42             ↪ R.id.loginPass)).perform(typeText("").toString()));
41         closeSoftKeyboard();
42         onView(withId(R.id.loginBtn)).perform(click());
43         onView(withText(R.string.passwordIsEmpty)).inRoot(withDecorView(not(is(
44             ↪ loginActivity.getWindow().getDecorView())))).check(matches(
45             ↪ isDisplayed()));
44     }
45 }
46
47 @Test
48 public void passwordIsTooSmall() {
49     if(auth.getCurrentUser() == null) {
50         auth.signInWithEmailAndPassword(onView(withId(R.id.loginEmail)).perform(
51             ↪ typeText("test@dataport.de").toString(), Espresso.onView(withId(
52             ↪ R.id.loginPass)).perform(typeText("1234").toString()));
51         closeSoftKeyboard();
52         onView(withId(R.id.loginBtn)).perform(click());
53         onView(withText(R.string.passwordIsTooSmall)).inRoot(withDecorView(not(
54             ↪ is(loginActivity.getWindow().getDecorView())))).check(matches(
55             ↪ isDisplayed()));
54     }
55 }
56
57 @Test
58 public void emailIsRequired() {
59     if(auth.getCurrentUser() == null) {
60         auth.signInWithEmailAndPassword(onView(withId(R.id.loginEmail)).perform(
61             ↪ typeText(" ").toString(), Espresso.onView(withId(R.id.loginPass))
62             ↪ .perform(typeText("123456").toString()));
61         closeSoftKeyboard();
62         onView(withId(R.id.loginBtn)).perform(click());
63         onView(withText(R.string.emailIsRequired)).inRoot(withDecorView(not(is(
64             ↪ loginActivity.getWindow().getDecorView())))).check(matches(
65             ↪ isDisplayed()));
64     }
65 }
66
67 @Test
68 public void emailIsInvalid() {

```

```
69     if(auth.getCurrentUser() == null) {
70         auth.signInWithEmailAndPassword(onView(withId(R.id.loginEmail)).perform(
71             ↪ typeText("blabla").toString(), Espresso.onView(withId(R.id.
72             ↪ loginPass)).perform(typeText("123456").toString()));
73         closeSoftKeyboard();
74         onView(withId(R.id.loginBtn)).perform(click());
75         onView(withId(R.string.emailIsInvalid)).inRoot(withDecorView(not(is(
76             ↪ loginActivity.getWindow().getDecorView())))).check(matches(
77             ↪ isDisplayed()));
78     }
79 }
80
81 @After
82 public void tearDown() throws Exception {
83     loginActivity = null;
84     if(auth != null) {
85         auth.signOut();
86         auth = null;
87     }
88 }
```

Code-Anhang 6: LoginTests

MainpageTests

```
1 public class MainpageTest {
2
3     private FirebaseAuth auth;
4
5     @Rule
6     public ActivityTestRule<Mainpage> mMainpageTestRule = new ActivityTestRule<>(
7         ↪ Mainpage.class);
8     private Mainpage mainpageActivity = null;
9     private Instrumentation.ActivityMonitor monitorProfile = getInstrumentation().
10         ↪ addMonitor(Profile.class.getName(), null, false);
11
12     @Before
13     public void setUp() throws Exception {
14         mainpageActivity = mMainpageTestRule.getActivity();
15         auth = FirebaseAuth.getInstance();
16     }
17
18     @Test
19     public void goToProfile(){
20         if(auth.getCurrentUser() != null) {
21             onView(withId(R.id.cardViewProfile)).perform(click());
22             Activity activity = getInstrumentation().waitForMonitorWithTimeout(
23                 ↪ monitorProfile, 2500);
24             assertNotNull(activity);
25             activity.finish();
26         }
27     }
28
29     @Test
30     public void logOut(){
31         if(auth.getCurrentUser() != null) {
32             onView(withId(R.id.logoutBtn)).perform(click());
33         }
34     }
35 }
```



```
30         Activity activity = getInstrumentation().waitForMonitorWithTimeout(  
31             ↪ monitorStartapp, 2500);  
32         assertNotNull(activity);  
33         activity.finish();  
34     }  
35 }  
36 @After  
37 public void tearDown() throws Exception {  
38     mainpageActivity = null;  
39 }  
40  
41 }
```

Code-Anhang 7: MainpageTests

InviteMembersTests

```
1 public class InviteMembersTest {  
2  
3     private FirebaseAuth auth;  
4  
5     @Rule  
6     public ActivityTestRule<InviteMembers> mInviteMembersTestRule = new  
7         ↪ ActivityTestRule<>(InviteMembers.class);  
8     private Instrumentation.ActivityMonitor monitorMainpage = getInstrumentation().  
9         ↪ addMonitor(Mainpage.class.getName(), null, false);  
10    private Instrumentation.ActivityMonitor monitorCreateEvent = getInstrumentation  
11        ↪ ().addMonitor(CreateEvent.class.getName(), null, false);  
12    private InviteMembers InviteMembersActivity = null;  
13  
14    @Before  
15    public void setUp() throws Exception {  
16        InviteMembersActivity = mInviteMembersTestRule.getActivity();  
17        auth = FirebaseAuth.getInstance();  
18    }  
19  
20    @Test  
21    public void testFAB() {  
22        if(auth.getCurrentUser() != null) {  
23            onView(withId(R.id.fabOpenInviteMembers)).perform(click());  
24        }  
25    }  
26  
27    @Test  
28    public void nobodyInvited() {  
29        if(auth.getCurrentUser() != null) {  
30            closeSoftKeyboard();  
31            onView(withId(R.id.inviteMembersBtn)).perform(click());  
32            onView(withText(R.string.nobodyInvited)).inRoot(withDecorView(not(is(  
33                ↪ InviteMembersActivity.getWindow().getDecorView())))).check(  
34                ↪ matches(isDisplayed()));  
35            Activity inviteMembersActivity = getInstrumentation().  
36                ↪ waitForMonitorWithTimeout(monitorMainpage, 2500);  
37            assertNotNull(inviteMembersActivity);  
38            inviteMembersActivity.finish();  
39        }  
40    }  
41 }
```

```
36     @Test
37     public void inviteSomeone() {
38         if(auth.getCurrentUser() != null) {
39             onView(withId(R.id.inviteMembersSearch)).perform(typeText("Joh"));
40             onData(anything()).inAdapterView(withId(R.id.inviteMembersListView)).
41                 ↪ atPosition(0).perform(click());
42             onData(anything()).inAdapterView(withId(R.id.inviteMembersListView)).
43                 ↪ atPosition(1).perform(click());
44             closeSoftKeyboard();
45             onView(withId(R.id.inviteMembersBtn)).perform(click());
46             Activity inviteMembersActivity = getInstrumentation().
47                 ↪ waitForMonitorWithTimeout(monitorCreateEvent, 2500);
48             assertNotNull(inviteMembersActivity);
49             inviteMembersActivity.finish();
50         }
51     }
52
53     @After
54     public void tearDown() throws Exception {
55         InviteMembersActivity = null;
56     }
57 }
```

Code-Anhang 8: InviteMembersTests

EventsTests

```
1 public class EventsTest {
2
3     private FirebaseAuth auth;
4     private FirebaseUser user;
5
6     @Rule
7     public ActivityTestRule<Events> mEventsTestRule = new ActivityTestRule<>(Events.
8         ↪ class);
9     private Instrumentation.ActivityMonitor monitorMainpage = getInstrumentation().
10        ↪ addMonitor(Mainpage.class.getName(), null, false);
11     private Instrumentation.ActivityMonitor monitorEvents = getInstrumentation().
12        ↪ addMonitor(Events.class.getName(), null, false);
13     private Instrumentation.ActivityMonitor monitorEditEvent = getInstrumentation().
14        ↪ addMonitor(EditEvent.class.getName(), null, false);
15     private Events eventsActivity = null;
16
17     @Before
18     public void setUp() throws Exception {
19         eventsActivity = mEventsTestRule.getActivity();
20         auth = FirebaseAuth.getInstance();
21         user = auth.getCurrentUser();
22     }
23
24     @Test
25     public void testFAB() {
26         if(auth.getCurrentUser() != null) {
27             onView(withId(R.id.fabOpenEvents)).perform(click());
28         }
29     }
30
31     @Test
```

```
28 public void searchEvent() {
29     if(auth.getCurrentUser() != null) {
30         onView(withId(R.id.eventsSearch)).perform(typeText("Test"));
31         onData(anything()).inAdapterView(withId(R.id.listView)).atPosition(0).
32             ↪ perform(click());
33         Activity inviteMembersActivity = getInstrumentation().
34             ↪ waitForMonitorWithTimeout(monitorEditEvent, 2500);
35         assertNotNull(inviteMembersActivity);
36         inviteMembersActivity.finish();
37     }
38 }
39
40 @Test
41 public void adminToTheEvent() {
42     if(auth.getCurrentUser() != null) {
43         onView(withId(R.id.eventsSearch)).perform(typeText("Test"));
44         onData(anything()).inAdapterView(withId(R.id.listView)).atPosition(0).
45             ↪ perform(click());
46         Activity inviteMembersActivity = getInstrumentation().
47             ↪ waitForMonitorWithTimeout(monitorEditEvent, 2500);
48         String email = user.getEmail();
49         if(email != null) onView(withId(R.id.editEventCreator)).check(matches(
50             ↪ withText(email)));
51         assertNotNull(inviteMembersActivity);
52         inviteMembersActivity.finish();
53     }
54 }
55
56 @Test
57 public void notAdminToTheEvent() {
58     if(auth.getCurrentUser() != null) {
59         onView(withId(R.id.eventsSearch)).perform(typeText("TestT"));
60         onData(anything()).inAdapterView(withId(R.id.listView)).atPosition(0).
61             ↪ perform(click());
62         Activity inviteMembersActivity = getInstrumentation().
63             ↪ waitForMonitorWithTimeout(monitorEditEvent, 2500);
64         String email = user.getEmail();
65         if(email != null) assertEquals(onView(withId(R.id.editEventCreator)).
66             ↪ toString(), email);
67         assertNotNull(inviteMembersActivity);
68         inviteMembersActivity.finish();
69     }
70 }
71
72 @Test
73 public void editEvent() {
74     onView(withId(R.id.eventsSearch)).perform(typeText("Test"));
75     onData(anything()).inAdapterView(withId(R.id.listView)).atPosition(0).
76         ↪ perform(click());
77     Activity inviteMembersActivity = getInstrumentation().
78         ↪ waitForMonitorWithTimeout(monitorEditEvent, 2500);
79     assertNotNull(inviteMembersActivity);
80     onView(withId(R.id.editEventPlaceText)).perform(clearText(), typeText("
81         ↪ TestEditEventPlace"));
82     closeSoftKeyboard();
83     onView(withId(R.id.editEventInfoText)).perform(clearText(), typeText("
84         ↪ TestEditEventInfo"));
85     closeSoftKeyboard();
86     onView(withId(R.id.editEventSaveBtn)).perform(click());
87     Activity editEventActivity = getInstrumentation().waitForMonitorWithTimeout(
88         ↪ monitorMainpage, 2500);
```

```
76     assertNotNull(editEventActivity);
77     editEventActivity.finish();
78 }
79
80 @Test
81 public void removeEvent() {
82     onView(withId(R.id.eventsSearch)).perform(typeText("TestRemoveMe"));
83     onData(anything()).inAdapterView(withId(R.id.listView)).atPosition(0).
84         ↪ perform(click());
85     Activity inviteMembersActivity = getInstrumentation().
86         ↪ waitForMonitorWithTimeout(monitorEditEvent, 2500);
87     assertNotNull(inviteMembersActivity);
88     closeSoftKeyboard();
89     onView(withId(R.id.editEventDeleteBtn)).perform(click());
90     onView(withText("Yes")).perform(click());
91     Activity editEventActivity = getInstrumentation().waitForMonitorWithTimeout(
92         ↪ monitorEvents, 2500);
93     assertNotNull(editEventActivity);
94     onView(withId(R.id.eventsSearch)).perform(typeText("TestRemoveMe"));
95     if(eventsActivity.getCountAdapter() == 0){
96         closeSoftKeyboard();
97         pressBack();
98     }
99 }
100
101 @After
102 public void tearDown() throws Exception {
103     eventsActivity = null;
104 }
```

Code-Anhang 9: EventsTests

SearchTests

```
1 public class SearchTest {
2
3     private FirebaseAuth auth;
4
5     @Rule
6     public ActivityTestRule<Search> mSearchTestRule = new ActivityTestRule<>(Search.
7         ↪ class);
8     private Instrumentation.ActivityMonitor monitorMainpage = getInstrumentation().
9         ↪ addMonitor(Mainpage.class.getName(), null, false);
10    private Instrumentation.ActivityMonitor monitorUserInfo = getInstrumentation().
11        ↪ addMonitor(UserInfo.class.getName(), null, false);
12    private Search searchActivity = null;
13
14    @Before
15    public void setUp() throws Exception {
16        searchActivity = mSearchTestRule.getActivity();
17        auth = FirebaseAuth.getInstance();
18    }
19
20    @Test
21    public void searchUser() {
22        if(auth.getCurrentUser() != null) {
23            onView(withId(R.id.searchUser)).perform(typeText("bob"));
24        }
25    }
26 }
```

```
21         onData(anything()).inAdapterView(withId(R.id.searchUserListView)).
22             ↪ atPosition(0).perform(click());
23         Activity userInfoActivity = getInstrumentation().
24             ↪ waitForMonitorWithTimeout(monitorUserInfo, 2500);
25         assertNotNull(userInfoActivity);
26         userInfoActivity.finish();
27     }
28     @Test
29     public void searchNotFoundUser() {
30         if(auth.getCurrentUser() != null) {
31             onView(withId(R.id.searchUser)).perform(typeText("bobby"));
32             if(searchActivity.getSizeAdapter() == 0){
33                 closeSoftKeyboard();
34                 pressBack();
35             }
36         }
37     }
38
39
40     @After
41     public void tearDown() throws Exception {
42         searchActivity = null;
43     }
44
45 }
```

Code-Anhang 10: SearchTests

Startapp.xml

```
1 <RelativeLayout
2
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     android:background="@drawable/bg"
9     tools:context="com.example.tonyk.promac.Startapp">
10
11     <ImageView
12         android:id="@+id/logoDP"
13         android:layout_width="240dp"
14         android:layout_height="90dp"
15         android:layout_below="@+id/welcomeText"
16         android:layout_centerHorizontal="true"
17         android:layout_marginTop="15dp"
18         android:contentDescription="@string/logoDP"
19         app:srcCompat="@drawable/logo_dp" />
20
21     <ImageButton
22         android:id="@+id/goToRegistration"
23         android:layout_width="160dp"
24         android:layout_height="160dp"
25         android:layout_below="@+id/startAppText"
26         android:layout_marginStart="21dp"
27         android:layout_marginTop="16dp"
```

```
28         android:background="@drawable/ripple_effect"
29         android:contentDescription="@string/goToRegistration"
30         android:scaleType="fitCenter"
31         app:srcCompat="@drawable/register" />
32
33     <ImageButton
34         android:id="@+id/goToLogin"
35         android:layout_width="160dp"
36         android:layout_height="160dp"
37         android:layout_alignParentEnd="true"
38         android:layout_below="@+id/startAppText"
39         android:layout_marginEnd="21dp"
40         android:layout_marginTop="16dp"
41         android:background="@drawable/ripple_effect"
42         android:contentDescription="@string/goToLogin"
43         android:scaleType="fitCenter"
44         app:srcCompat="@drawable/login" />
45
46 </RelativeLayout>
```

Code-Anhang 11: Start der App

Login.xml

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:id="@+id/loginForm"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:background="@drawable/login_bg"
7     android:clickable="true"
8     android:focusable="true"
9     android:focusableInTouchMode="true"
10    tools:context="com.example.tonyk.promac.Login">
11
12    <RelativeLayout
13        android:layout_width="300dp"
14        android:layout_height="210dp"
15        android:layout_centerHorizontal="true"
16        android:layout_centerVertical="true">
17
18        <TextView
19            android:id="@+id/loginText"
20            android:layout_width="wrap_content"
21            android:layout_height="wrap_content"
22            android:layout_alignParentTop="true"
23            android:layout_centerHorizontal="true"
24            android:layout_marginTop="12dp"
25            android:text="@string/loginText"
26            android:textColor="@android:color/holo_blue_dark"
27            android:textSize="27sp"
28            android:textStyle="bold" />
29
30        <EditText
31            android:id="@+id/loginEmail"
32            android:layout_width="230dp"
33            android:layout_height="48dp"
34            android:layout_alignStart="@+id/loginPass"
35            android:layout_below="@+id/loginText"
```

```
36         android:layout_marginTop="8dp"
37         android:ems="10"
38         android:hint="@string/loginEmail"
39         android:inputType="textEmailAddress" />
40
41     <EditText
42         android:id="@+id/loginPass"
43         android:layout_width="230dp"
44         android:layout_height="48dp"
45         android:layout_alignEnd="@+id/loginText"
46         android:layout_alignTop="@+id/loginEmail"
47         android:layout_marginEnd="8dp"
48         android:layout_marginTop="38dp"
49         android:ems="10"
50         android:hint="@string/loginPass"
51         android:inputType="textPassword" />
52
53     <Button
54         android:id="@+id/loginBtn"
55         android:layout_width="wrap_content"
56         android:layout_height="wrap_content"
57         android:layout_below="@+id/loginPass"
58         android:layout_centerHorizontal="true"
59         android:layout_marginTop="7dp"
60         android:background="@drawable/ripple_effect"
61         android:text="@string/loginBtn"
62         android:textColor="#fff" />
63
64     <ProgressBar
65         android:id="@+id/loginProgressBar"
66         android:layout_width="wrap_content"
67         android:layout_height="wrap_content"
68         android:layout_centerHorizontal="true"
69         android:layout_centerVertical="true"
70         android:text="@string/loginProgressBar"
71         android:visibility="gone" />
72
73 </RelativeLayout>
74
75 </RelativeLayout>
```

Code-Anhang 12: AnmeldungDesign

Mainpage.xml

```
1 <LinearLayout
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:id="@+id/mainpage"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     android:background="@drawable/bg"
9     android:clickable="true"
10    android:focusable="true"
11    android:focusableInTouchMode="true"
12    android:orientation="vertical"
13    tools:context="com.example.tonyk.promac.Mainpage">
14
```

```
15 <RelativeLayout
16     android:layout_width="match_parent"
17     android:layout_height="0dp"
18     android:layout_weight="2">
19
20     <ImageButton
21         android:id="@+id/logoutBtn"
22         android:layout_width="60dp"
23         android:layout_height="60dp"
24         android:layout_above="@+id/mainpageText"
25         android:layout_alignParentEnd="true"
26         android:layout_marginBottom="12dp"
27         android:layout_marginEnd="12dp"
28         android:background="@drawable/ripple_effect"
29         android:src="@drawable/ic_exit_to_app_white_24dp"
30         android:contentDescription="@string/logoutBtn"
31         android:textColor="#fff" />
32
33 </RelativeLayout>
34
35 <GridLayout
36     android:id="@+id/mainGrid"
37     android:layout_width="match_parent"
38     android:layout_height="0dp"
39     android:layout_marginBottom="12dp"
40     android:layout_weight="8"
41     android:alignmentMode="alignMargins"
42     android:columnCount="2"
43     android:columnOrderPreserved="false"
44     android:padding="14dp"
45     android:rowCount="3">
46
47     <!-- Row 1, Column 1 -->
48     <android.support.v7.widget.CardView
49         android:id="@+id/cardViewProfile"
50         android:layout_width="0dp"
51         android:layout_height="0dp"
52         android:layout_columnWeight="1"
53         android:layout_marginBottom="16dp"
54         android:layout_marginEnd="6dp"
55         android:layout_rowWeight="1"
56         app:cardBackgroundColor="#48000000"
57         app:cardCornerRadius="8dp"
58         app:cardElevation="8dp">
59
60         <LinearLayout
61             android:layout_width="wrap_content"
62             android:layout_height="wrap_content"
63             android:layout_gravity="center_horizontal|center_vertical"
64             android:orientation="vertical">
65
66             <ImageView
67                 android:layout_width="wrap_content"
68                 android:layout_height="wrap_content"
69                 android:layout_gravity="center_horizontal"
70                 android:contentDescription="@string/cardImageProfile"
71                 android:src="@drawable/ic_person_pin_white_24dp" />
72
73             <TextView
74                 android:layout_width="wrap_content"
75                 android:layout_height="wrap_content"
```



```
76         android:text="@string/cardProfile"
77         android:textAlignment="center"
78         android:textColor="#fff"
79         android:textSize="18sp"
80         android:textStyle="bold" />
81
82     </LinearLayout>
83
84 </android.support.v7.widget.CardView>
85
86 </GridLayout>
87
88 </LinearLayout>
```

Code-Anhang 13: MainpageDesign

ProfileContent.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.constraint.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:background="@drawable/bg"
8     tools:context="com.example.tonyk.promac.Profile"
9     tools:showIn="@layout/activity_profile">
10
11 </android.support.constraint.ConstraintLayout>
```

Code-Anhang 14: ProfileContentDesign

Profile.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout
3
4     xmlns:android="http://schemas.android.com/apk/res/android"
5     xmlns:app="http://schemas.android.com/apk/res-auto"
6     xmlns:tools="http://schemas.android.com/tools"
7     android:layout_width="match_parent"
8     android:layout_height="match_parent"
9     android:background="@drawable/bg">
10
11     <include layout="@layout/content_profile" />
12
13     <RelativeLayout
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content"
16         android:clickable="true"
17         android:focusable="true"
18         android:focusableInTouchMode="true">
19
20         <android.support.design.widget.CoordinatorLayout
21             xmlns:android="http://schemas.android.com/apk/res/android"
22             xmlns:tools="http://schemas.android.com/tools">
```

```
23     android:layout_width="match_parent"
24     android:layout_height="match_parent"
25     tools:context="com.example.tonyk.promac.Profile">
26
27     <android.support.constraint.ConstraintLayout
28         android:layout_width="match_parent"
29         android:layout_height="700dp">
30
31         <View
32             android:id="@+id/view"
33             android:layout_width="0dp"
34             android:layout_height="155dp"
35             android:layout_marginTop="68dp"
36             android:background="#38000000"
37             android:contentDescription="@string/profileInfoField"
38             app:layout_constraintEnd_toStartOf="@+id/guideline7"
39             app:layout_constraintHorizontal_bias="0.0"
40             app:layout_constraintStart_toStartOf="@+id/guideline6"
41             app:layout_constraintTop_toTopOf="parent" />
42
43         <ImageView
44             android:id="@+id/userInfoPhoto"
45             android:layout_width="0dp"
46             android:layout_height="108dp"
47             android:contentDescription="@string/profileInfoPhoto"
48             app:layout_constraintBottom_toTopOf="@+id/view"
49             app:layout_constraintDimensionRatio="w,1:1"
50             app:layout_constraintEnd_toStartOf="@+id/guideline7"
51             app:layout_constraintStart_toStartOf="@+id/guideline6"
52             app:layout_constraintTop_toTopOf="@+id/view"
53             app:srcCompat="@drawable/ic_person_pin_white_24dp" />
54
55         <EditText
56             android:id="@+id/userInfoName"
57             android:layout_width="240dp"
58             android:layout_height="wrap_content"
59             android:layout_marginEnd="8dp"
60             android:layout_marginStart="8dp"
61             android:layout_marginTop="21dp"
62             android:inputType="textPersonName"
63             android:labelFor="@+id/userInfoName"
64             android:text="@string/profileInfoName"
65             android:textAlignment="center"
66             android:textColor="@android:color/white"
67             android:textSize="14sp"
68             android:textStyle="bold"
69             app:layout_constraintEnd_toEndOf="@+id/userInfoPhoto"
70             app:layout_constraintStart_toStartOf="@+id/userInfoPhoto"
71             app:layout_constraintTop_toBottomOf="@+id/userInfoPhoto" />
72
73         <TextView
74             android:id="@+id/userInfoEmail"
75             android:layout_width="240dp"
76             android:layout_height="wrap_content"
77             android:layout_marginEnd="8dp"
78             android:layout_marginTop="10dp"
79             android:text="@string/profileInfoEmail"
80             android:textAlignment="center"
81             android:textColor="@android:color/white"
82             android:textSize="16sp"
83             android:textStyle="italic"
```

```
84         app:layout_constraintEnd_toEndOf="@+id/userInfoName"
85         app:layout_constraintHorizontal_bias="0.0"
86         app:layout_constraintStart_toStartOf="@+id/userInfoName"
87         app:layout_constraintTop_toBottomOf="@+id/userInfoName" />
88
89     <ImageButton
90         android:id="@+id/uploadBtn"
91         android:layout_width="50dp"
92         android:layout_height="50dp"
93         android:layout_marginStart="36dp"
94         android:layout_marginTop="24dp"
95         android:background="@drawable/ripple_effect"
96         android:contentDescription="@string/uploadBtn"
97         android:src="@drawable/ic_file_upload_white_24dp"
98         android:text="@string/uploadBtn"
99         android:textColor="#fff"
100        android:visibility="invisible"
101        app:layout_constraintEnd_toStartOf="@+id/userInfoPhoto"
102        app:layout_constraintHorizontal_bias="0.026"
103        app:layout_constraintStart_toStartOf="@+id/guideline6"
104        app:layout_constraintTop_toBottomOf="@+id/userInfoAdditional"
105        app:layout_constraintTop_toTopOf="@+id/userInfoPhoto"
106        tools:ignore="VectorDrawableCompat" />
107
108    <TextView
109        android:id="@+id/uploadPhoto"
110        android:layout_width="100dp"
111        android:layout_height="45dp"
112        android:layout_marginStart="10dp"
113        android:layout_marginTop="82dp"
114        android:lines="2"
115        android:text="@string/uploadPhoto"
116        android:textAlignment="center"
117        android:textColor="#fff"
118        android:textSize="14sp"
119        android:visibility="invisible"
120        app:layout_constraintEnd_toStartOf="@+id/userInfoPhoto"
121        app:layout_constraintHorizontal_bias="0.026"
122        app:layout_constraintStart_toStartOf="@+id/guideline6"
123        app:layout_constraintTop_toBottomOf="@+id/userInfoAdditional"
124        app:layout_constraintTop_toTopOf="@+id/userInfoPhoto"
125        tools:ignore="VectorDrawableCompat" />
126
127    <ProgressBar
128        android:id="@+id/profileProgressBar"
129        android:layout_width="157dp"
130        android:layout_height="match_parent"
131        android:layout_marginTop="62dp"
132        android:text="@string/profileProgressBar"
133        android:visibility="gone"
134        app:layout_constraintBottom_toTopOf="@+id/view"
135        app:layout_constraintDimensionRatio="w,1:1"
136        app:layout_constraintEnd_toStartOf="@+id/guideline7"
137        app:layout_constraintStart_toStartOf="@+id/guideline6"
138        app:layout_constraintTop_toTopOf="@+id/view"
139        tools:ignore="VectorDrawableCompat" />
140
141    <View
142        android:id="@+id/userInfoLine"
143        android:layout_width="0dp"
144        android:layout_height="2dp"
```

```

145         android:layout_marginTop="5dp"
146         android:background="@color/red_dp"
147         app:layout_constraintEnd_toEndOf="parent"
148         app:layout_constraintStart_toStartOf="parent"
149         app:layout_constraintTop_toBottomOf="@+id/userInfoText" />
150
151     <TextView
152         android:id="@+id/userInfoDepartment"
153         android:layout_width="wrap_content"
154         android:layout_height="wrap_content"
155         android:layout_marginTop="20dp"
156         android:text="@string/profileInfoDepartment"
157         android:textColor="@android:color/white"
158         android:textSize="18sp"
159         android:textStyle="bold"
160         app:layout_constraintStart_toStartOf="@+id/guideline6"
161         app:layout_constraintTop_toTopOf="@+id/userInfoLine" />
162
163     <EditText
164         android:id="@+id/userInfoDepartmentText"
165         android:layout_width="240dp"
166         android:layout_height="wrap_content"
167         android:layout_marginStart="20dp"
168         android:ellipsize="end"
169         android:hint="@string/profileInfoDepartmentText"
170         android:maxLength="50"
171         android:maxLines="1"
172         android:textColor="@android:color/white"
173         android:textColorHint="#fff"
174         android:textSize="16sp"
175         app:layout_constraintBaseline_toBaselineOf="@+id/
            ↪ userInfoDepartment"
176         app:layout_constraintStart_toEndOf="@+id/userInfoDepartment" />
177
178 </android.support.constraint.ConstraintLayout>
179
180 <android.support.design.widget.FloatingActionButton
181     android:id="@+id/fabOpenProfile"
182     android:layout_width="wrap_content"
183     android:layout_height="wrap_content"
184     android:layout_gravity="bottom|end"
185     android:layout_marginBottom="25dp"
186     android:layout_marginEnd="@dimen/fab_margin"
187     android:theme="@style/Theme.Design.Light"
188     android:src="@drawable/ic_add_white_36dp" />
189
190 <android.support.design.widget.FloatingActionButton
191     android:id="@+id/fabProfileToEditEvent"
192     android:layout_width="wrap_content"
193     android:layout_height="wrap_content"
194     android:layout_gravity="bottom|end"
195     android:layout_marginBottom="156dp"
196     android:layout_marginEnd="@dimen/fab_margin"
197     android:theme="@style/Theme.Design.Light"
198     android:src="@drawable/edit_event"
199     android:visibility="invisible" />
200
201 <ImageButton
202     android:id="@+id/profileSaveBtn"
203     android:layout_width="65dp"
204     android:layout_height="65dp"

```

```
205         android:layout_gravity="bottom|start"
206         android:layout_marginBottom="25dp"
207         android:layout_marginStart="@dimen/fab_margin"
208         android:contentDescription="@string/profileSaveBtn"
209         android:background="@drawable/ripple_effect"
210         android:src="@drawable/ic_save_white_24dp"
211         android:text="@string/profileSaveBtn"
212         android:textColor="#fff"
213         app:layout_constraintEnd_toStartOf="@+id/guideline7"
214         app:layout_constraintHorizontal_bias="0.0"
215         app:layout_constraintStart_toStartOf="@+id/guideline6"
216         app:layout_constraintTop_toBottomOf="@+id/userInfoAdditional" />
217
218     </android.support.design.widget.CoordinatorLayout>
219
220 </RelativeLayout>
221
222 </RelativeLayout>
```

Code-Anhang 15: ProfileDesign

CreateEvent.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout
3
4     xmlns:android="http://schemas.android.com/apk/res/android"
5     xmlns:app="http://schemas.android.com/apk/res-auto"
6     xmlns:tools="http://schemas.android.com/tools"
7     android:layout_width="match_parent"
8     android:layout_height="match_parent"
9     android:background="@drawable/bg">
10
11     <include layout="@layout/content_create_event" />
12
13     <RelativeLayout
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content"
16         android:clickable="true"
17         android:focusable="true"
18         android:focusableInTouchMode="true">
19
20         <android.support.design.widget.CoordinatorLayout xmlns:android="http://
21             ↳ schemas.android.com/apk/res/android"
22             xmlns:tools="http://schemas.android.com/tools"
23             android:id="@+id/createEventLayout"
24             android:layout_width="match_parent"
25             android:layout_height="match_parent"
26             tools:context="com.example.tonyk.promac.CreateEvent">
27
28             <android.support.constraint.ConstraintLayout
29                 android:layout_width="match_parent"
30                 android:layout_height="700dp">
31
32                 <android.support.constraint.Guideline
33                     android:id="@+id/guideline6"
34                     android:layout_width="wrap_content"
35                     android:layout_height="wrap_content"
```

```
36         android:orientation="vertical"
37         app:layout_constraintGuide_percent="0.052083332" />
38
39     <android.support.constraint.Guideline
40         android:id="@+id/guideline7"
41         android:layout_width="wrap_content"
42         android:layout_height="wrap_content"
43         android:orientation="vertical"
44         app:layout_constraintGuide_percent="0.9557292" />
45
46     <View
47         android:id="@+id/view"
48         android:layout_width="0dp"
49         android:layout_height="136dp"
50         android:layout_marginTop="18dp"
51         android:background="#38000000"
52         android:contentDescription="@string/profileInfoField"
53         app:layout_constraintEnd_toStartOf="@+id/guideline7"
54         app:layout_constraintHorizontal_bias="0.0"
55         app:layout_constraintStart_toStartOf="@+id/guideline6"
56         app:layout_constraintTop_toTopOf="parent" />
57
58     <EditText
59         android:id="@+id/createEventName"
60         android:layout_width="270dp"
61         android:layout_height="36dp"
62         android:layout_marginEnd="8dp"
63         android:layout_marginStart="8dp"
64         android:layout_marginTop="21dp"
65         android:hint="@string/createEventName"
66         android:inputType="textPersonName"
67         android:labelFor="@+id/createEventName"
68         android:maxLength="30"
69         android:textAlignment="center"
70         android:textColor="@android:color/white"
71         android:textColorHint="#000"
72         android:textSize="14sp"
73         app:layout_constraintEnd_toStartOf="@+id/guideline7"
74         app:layout_constraintStart_toStartOf="@+id/guideline6"
75         app:layout_constraintTop_toTopOf="parent" />
76
77 </android.support.constraint.ConstraintLayout>
78
79 <ImageButton
80     android:id="@+id/createEventSaveBtn"
81     android:layout_width="65dp"
82     android:layout_height="65dp"
83     android:layout_gravity="bottom|start"
84     android:layout_marginBottom="25dp"
85     android:layout_marginStart="@dimen/fab_margin"
86     android:background="@drawable/ripple_effect"
87     android:contentDescription="@string/profileSaveBtn"
88     android:src="@drawable/ic_save_white_24dp"
89     android:text="@string/profileSaveBtn"
90     android:textColor="#fff"
91     app:layout_constraintEnd_toStartOf="@+id/guideline7"
92     app:layout_constraintHorizontal_bias="0.026"
93     app:layout_constraintStart_toStartOf="@+id/guideline6"
94     app:layout_constraintTop_toBottomOf="@+id/createEventInfo" />
95
96 </android.support.design.widget.CoordinatorLayout>
```

```
97
98     </RelativeLayout>
99
100 </RelativeLayout>
```

Code-Anhang 16: CreateEventDesign

Events.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout
3
4     xmlns:android="http://schemas.android.com/apk/res/android"
5     xmlns:app="http://schemas.android.com/apk/res-auto"
6     xmlns:tools="http://schemas.android.com/tools"
7     android:layout_width="match_parent"
8     android:layout_height="match_parent"
9     android:background="@drawable/bg">
10
11     <include layout="@layout/content_events" />
12
13     <RelativeLayout
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content"
16         android:clickable="true"
17         android:focusable="true"
18         android:focusableInTouchMode="true">
19
20         <android.support.design.widget.CoordinatorLayout
21             xmlns:android="http://schemas.android.com/apk/res/android"
22             xmlns:tools="http://schemas.android.com/tools"
23             android:layout_width="match_parent"
24             android:layout_height="match_parent"
25             tools:context="com.example.tonyk.promac.Events">
26
27             <android.support.constraint.ConstraintLayout
28                 android:layout_width="match_parent"
29                 android:layout_height="689dp">
30
31
32                 <SearchView
33                     android:id="@+id/eventsSearch"
34                     android:layout_width="320dp"
35                     android:layout_height="50dp"
36                     android:layout_marginBottom="8dp"
37                     android:layout_marginEnd="8dp"
38                     android:layout_marginStart="8dp"
39                     android:layout_marginTop="8dp"
40                     android:clickable="true"
41                     android:focusable="true"
42                     android:background="#24000000"
43                     android:queryHint="@string/eventsSearchField"
44                     app:layout_constraintBottom_toTopOf="@+id/listView"
45                     app:layout_constraintEnd_toEndOf="parent"
46                     app:layout_constraintStart_toStartOf="parent"
47                     app:layout_constraintTop_toTopOf="parent" />
48
49                 <ListView
50                     android:id="@+id/listView"
```

```
51         style="@style/ListView"
52         android:layout_width="340dp"
53         android:layout_height="360dp"
54         android:layout_marginBottom="120dp"
55         android:background="@color/transp"
56         android:backgroundTint="#f0ffff"
57         android:drawSelectorOnTop="true"
58         app:layout_constraintBottom_toBottomOf="parent"
59         app:layout_constraintEnd_toEndOf="parent"
60         app:layout_constraintStart_toStartOf="parent"
61         app:layout_constraintTop_toTopOf="parent" />
62
63     </android.support.constraint.ConstraintLayout>
64
65 </android.support.design.widget.CoordinatorLayout>
66
67 </RelativeLayout>
68
69 </RelativeLayout>
```

Code-Anhang 17: Events

EditEvent.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout
3
4     xmlns:android="http://schemas.android.com/apk/res/android"
5     xmlns:app="http://schemas.android.com/apk/res-auto"
6     xmlns:tools="http://schemas.android.com/tools"
7     android:layout_width="match_parent"
8     android:layout_height="match_parent"
9     android:background="@drawable/bg">
10
11     <include layout="@layout/content_edit_event" />
12
13     <RelativeLayout
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content"
16         android:clickable="true"
17         android:focusable="true"
18         android:focusableInTouchMode="true">
19
20         <android.support.design.widget.CoordinatorLayout
21             xmlns:android="http://schemas.android.com/apk/res/android"
22             xmlns:tools="http://schemas.android.com/tools"
23             android:layout_width="match_parent"
24             android:layout_height="match_parent"
25             tools:context="com.example.tonyk.promac.EditEvent">
26
27             <android.support.constraint.ConstraintLayout
28                 android:layout_width="match_parent"
29                 android:layout_height="700dp">
30
31
32                 <android.support.constraint.Guideline
33                     android:id="@+id/guideline6"
34                     android:layout_width="wrap_content"
35                     android:layout_height="wrap_content"
```



```
36         android:orientation="vertical"
37         app:layout_constraintGuide_percent="0.052083332" />
38
39     <android.support.constraint.Guideline
40         android:id="@+id/guideline7"
41         android:layout_width="wrap_content"
42         android:layout_height="wrap_content"
43         android:orientation="vertical"
44         app:layout_constraintGuide_percent="0.9557292" />
45
46     <View
47         android:id="@+id/view"
48         android:layout_width="0dp"
49         android:layout_height="150dp"
50         android:layout_marginTop="18dp"
51         android:background="#38000000"
52         android:contentDescription="@string/profileInfoField"
53         app:layout_constraintEnd_toStartOf="@+id/guideline7"
54         app:layout_constraintHorizontal_bias="0.0"
55         app:layout_constraintStart_toStartOf="@+id/guideline6"
56         app:layout_constraintTop_toTopOf="parent" />
57
58     <TextView
59         android:id="@+id/editEventName"
60         android:layout_width="300dp"
61         android:layout_height="wrap_content"
62         android:layout_marginEnd="8dp"
63         android:layout_marginStart="8dp"
64         android:layout_marginTop="21dp"
65         android:labelFor="@+id/editEventName"
66         android:paddingTop="5dp"
67         android:text="@string/editEventName"
68         android:textAlignment="center"
69         android:textColor="@android:color/white"
70         android:textSize="14sp"
71         android:textStyle="bold"
72         app:layout_constraintEnd_toStartOf="@+id/guideline7"
73         app:layout_constraintStart_toStartOf="@+id/guideline6"
74         app:layout_constraintTop_toTopOf="parent" />
75
76     <TextView
77         android:id="@+id/editEventMembers"
78         android:layout_width="220dp"
79         android:layout_height="52dp"
80         android:layout_marginEnd="8dp"
81         android:layout_marginStart="13dp"
82         android:layout_marginTop="8dp"
83         android:ellipsize="end"
84         android:labelFor="@+id/editEventMembers"
85         android:maxLines="3"
86         android:scrollbars="vertical"
87         android:text="@string/editEventMembers"
88         android:textAlignment="center"
89         android:textColor="@android:color/white"
90         android:textSize="14sp"
91         android:textStyle="bold"
92         app:layout_constraintEnd_toStartOf="@+id/guideline7"
93         app:layout_constraintHorizontal_bias="0.482"
94         app:layout_constraintStart_toStartOf="@+id/guideline6"
95         app:layout_constraintTop_toBottomOf="@+id/editEventCreator" />
96
```

```
97         <EditText
98             android:id="@+id/editEventDeadline"
99             android:layout_width="220dp"
100             android:layout_height="35dp"
101             android:layout_marginEnd="8dp"
102             android:layout_marginStart="10dp"
103             android:layout_marginTop="8dp"
104             android:clickable="true"
105             android:editable="false"
106             android:focusable="true"
107             android:inputType="none"
108             android:labelFor="@+id/editEventDeadline"
109             android:paddingTop="3dp"
110             android:text="@string/editEventDeadline"
111             android:textAlignment="center"
112             android:textColor="@android:color/white"
113             android:textSize="14sp"
114             android:textStyle="bold"
115             app:layout_constraintEnd_toStartOf="@+id/guideline7"
116             app:layout_constraintStart_toStartOf="@+id/guideline6"
117             app:layout_constraintTop_toBottomOf="@+id/editEventMembers" />
118
119         <TextView
120             android:id="@+id/editEventCreator"
121             android:layout_width="300dp"
122             android:layout_height="wrap_content"
123             android:layout_marginEnd="8dp"
124             android:layout_marginTop="1dp"
125             android:text="@string/editEventCreator"
126             android:textAlignment="center"
127             android:textColor="@android:color/white"
128             android:textSize="16sp"
129             android:textStyle="italic"
130             app:layout_constraintEnd_toEndOf="@+id/editEventName"
131             app:layout_constraintHorizontal_bias="0.0"
132             app:layout_constraintStart_toStartOf="@+id/editEventName"
133             app:layout_constraintTop_toBottomOf="@+id/editEventName" />
134
135         <TextView
136             android:id="@+id/editEventText"
137             android:layout_width="0dp"
138             android:layout_height="wrap_content"
139             android:layout_marginTop="7dp"
140             android:gravity="start"
141             android:text="@string/editEventText"
142             android:textAlignment="textStart"
143             android:textColor="@android:color/white"
144             android:textSize="24sp"
145             android:textStyle="bold"
146             app:layout_constraintEnd_toStartOf="@+id/guideline7"
147             app:layout_constraintStart_toStartOf="@+id/guideline6"
148             app:layout_constraintTop_toBottomOf="@+id/view" />
149
150         <View
151             android:id="@+id/userInfoLine"
152             android:layout_width="0dp"
153             android:layout_height="2dp"
154             android:layout_marginTop="5dp"
155             android:background="@color/red_dp"
156             app:layout_constraintEnd_toEndOf="parent"
157             app:layout_constraintStart_toStartOf="parent"
```

```

158         app:layout_constraintTop_toBottomOf="@+id/editEventText" />
159
160     <TextView
161         android:id="@+id/editEventPurpose"
162         android:layout_width="wrap_content"
163         android:layout_height="wrap_content"
164         android:layout_marginStart="3dp"
165         android:layout_marginTop="18dp"
166         android:text="@string/editEventPurpose"
167         android:textColor="@android:color/white"
168         android:textSize="18sp"
169         android:textStyle="bold"
170         app:layout_constraintStart_toStartOf="@+id/guideline6"
171         app:layout_constraintTop_toTopOf="@+id/userInfoLine" />
172
173     <EditText
174         android:id="@+id/editEventPurposeText"
175         android:layout_width="240dp"
176         android:layout_height="wrap_content"
177         android:layout_marginStart="24dp"
178         android:hint="@string/editEventPurposeText"
179         android:lines="2"
180         android:maxLength="240"
181         android:maxLines="2"
182         android:minLines="1"
183         android:textColor="@android:color/white"
184         android:textColorHint="#000"
185         android:textSize="16sp"
186         app:layout_constraintBaseline_toBaselineOf="@+id/
187             ↪ editEventPurpose"
188         app:layout_constraintStart_toEndOf="@+id/editEventPurpose" />
189
190     <TextView
191         android:id="@+id/editEventPlan"
192         android:layout_width="wrap_content"
193         android:layout_height="wrap_content"
194         android:layout_marginStart="3dp"
195         android:layout_marginTop="24dp"
196         android:text="@string/editEventPlan"
197         android:textColor="@android:color/white"
198         android:textSize="18sp"
199         android:textStyle="bold"
200         app:layout_constraintStart_toStartOf="@+id/guideline6"
201         app:layout_constraintTop_toBottomOf="@+id/editEventPurpose" />
202
203     <EditText
204         android:id="@+id/editEventPlanText"
205         android:layout_width="240dp"
206         android:layout_height="wrap_content"
207         android:layout_marginStart="12dp"
208         android:hint="@string/editEventPlanText"
209         android:lines="2"
210         android:maxLength="1200"
211         android:maxLines="2"
212         android:minLines="1"
213         android:textColor="@android:color/white"
214         android:textColorHint="#000"
215         android:textSize="16sp"
216         app:layout_constraintBaseline_toBaselineOf="@+id/editEventPlan"
217         app:layout_constraintStart_toEndOf="@+id/editEventPlan" />

```

```
218         <TextView
219             android:id="@+id/editEventPlace"
220             android:layout_width="wrap_content"
221             android:layout_height="wrap_content"
222             android:layout_marginStart="3dp"
223             android:layout_marginTop="24dp"
224             android:text="@string/editEventPlace"
225             android:textColor="@android:color/white"
226             android:textSize="18sp"
227             android:textStyle="bold"
228             app:layout_constraintStart_toStartOf="@+id/guideline6"
229             app:layout_constraintTop_toBottomOf="@+id/editEventPlan" />
230
231         <EditText
232             android:id="@+id/editEventPlaceText"
233             android:layout_width="240dp"
234             android:layout_height="wrap_content"
235             android:layout_marginStart="48dp"
236             android:hint="@string/editEventPlaceText"
237             android:lines="2"
238             android:maxLength="120"
239             android:maxLines="2"
240             android:minLines="1"
241             android:textColor="@android:color/white"
242             android:textColorHint="#000"
243             android:textSize="16sp"
244             app:layout_constraintBaseline_toBaselineOf="@+id/editEventPlace"
245             app:layout_constraintStart_toEndOf="@+id/editEventPlace" />
246
247         <TextView
248             android:id="@+id/editEventPrice"
249             android:layout_width="wrap_content"
250             android:layout_height="wrap_content"
251             android:layout_marginStart="3dp"
252             android:layout_marginTop="24dp"
253             android:text="@string/editEventPrice"
254             android:textAlignment="center"
255             android:textColor="@android:color/white"
256             android:textSize="18sp"
257             android:textStyle="bold"
258             app:layout_constraintStart_toStartOf="@+id/guideline6"
259             app:layout_constraintTop_toBottomOf="@+id/editEventPlace" />
260
261         <EditText
262             android:id="@+id/editEventPriceText"
263             android:layout_width="240dp"
264             android:layout_height="wrap_content"
265             android:layout_marginStart="51dp"
266             android:hint="@string/editEventPriceText"
267             android:lines="2"
268             android:maxLength="120"
269             android:maxLines="2"
270             android:minLines="1"
271             android:textColor="@android:color/white"
272             android:textColorHint="#000"
273             android:textSize="16sp"
274             app:layout_constraintBaseline_toBaselineOf="@+id/editEventPrice"
275             app:layout_constraintStart_toEndOf="@+id/editEventPrice" />
276
277         <TextView
278             android:id="@+id/editEventInfo"
```

```
279         android:layout_width="wrap_content"
280         android:layout_height="wrap_content"
281         android:layout_marginStart="3dp"
282         android:layout_marginTop="24dp"
283         android:text="@string/editEventInfo"
284         android:textColor="@android:color/white"
285         android:textSize="18sp"
286         android:textStyle="bold"
287         app:layout_constraintStart_toStartOf="@+id/guideline6"
288         app:layout_constraintTop_toBottomOf="@+id/editEventPrice" />
289
290     <EditText
291         android:id="@+id/editEventInfoText"
292         android:layout_width="240dp"
293         android:layout_height="wrap_content"
294         android:layout_marginStart="46dp"
295         android:hint="@string/editEventInfoText"
296         android:lines="3"
297         android:maxLength="2400"
298         android:maxLines="3"
299         android:minLines="1"
300         android:textColor="@android:color/white"
301         android:textColorHint="#000"
302         android:textSize="16sp"
303         app:layout_constraintBottom_toBottomOf="@+id/editEventInfo"
304         app:layout_constraintStart_toEndOf="@+id/editEventInfo" />
305
306     <ImageButton
307         android:id="@+id/editEventInviteMembersBtn"
308         android:layout_width="40dp"
309         android:layout_height="40dp"
310         android:layout_marginBottom="7dp"
311         android:layout_marginEnd="8dp"
312         android:layout_marginStart="8dp"
313         android:visibility="invisible"
314         android:background="@drawable/ripple_effect"
315         android:contentDescription="@string/editEventInviteMembersBtn"
316         android:src="@drawable/ic_person_add_white_24dp"
317         android:text="@string/editEventInviteMembersBtn"
318         app:layout_constraintBottom_toTopOf="@+id/editEventDeadline"
319         app:layout_constraintEnd_toStartOf="@+id/guideline7"
320         app:layout_constraintStart_toEndOf="@+id/editEventMembers"
321         app:layout_constraintTop_toTopOf="@+id/editEventMembers" />
322
323 </android.support.constraint.ConstraintLayout>
324
325 <android.support.design.widget.FloatingActionButton
326     android:id="@+id/fabOpenEditEvent"
327     android:layout_width="wrap_content"
328     android:layout_height="wrap_content"
329     android:layout_gravity="bottom|end"
330     android:layout_marginBottom="25dp"
331     android:layout_marginEnd="@dimen/fab_margin"
332     android:theme="@style/Theme.Design.Light"
333     android:src="@drawable/ic_add_white_36dp" />
334
335 <android.support.design.widget.FloatingActionButton
336     android:id="@+id/fabEditEventToProfile"
337     android:layout_width="wrap_content"
338     android:layout_height="wrap_content"
339     android:layout_gravity="bottom|end"
```

```
340         android:layout_marginBottom="222dp"
341         android:layout_marginEnd="@dimen/fab_margin"
342         android:src="@drawable/ic_person_pin_white_24dp"
343         android:theme="@style/Theme.Design.Light"
344         android:visibility="invisible" />
345
346     <android.support.design.widget.FloatingActionButton
347     android:id="@+id/fabEditEventToMenu"
348     android:layout_width="wrap_content"
349     android:layout_height="wrap_content"
350     android:layout_gravity="bottom|end"
351     android:layout_marginBottom="@dimen/fab_margin"
352     android:layout_marginEnd="90dp"
353     android:src="@drawable/ic_dashboard_white_24dp"
354     android:theme="@style/Theme.Design.Light"
355     android:visibility="invisible" />
356
357     <android.support.design.widget.FloatingActionButton
358     android:id="@+id/fabEditEventToCreateEvent"
359     android:layout_width="wrap_content"
360     android:layout_height="wrap_content"
361     android:layout_gravity="bottom|end"
362     android:layout_marginBottom="156dp"
363     android:layout_marginEnd="@dimen/fab_margin"
364     android:src="@drawable/edit_event"
365     android:theme="@style/Theme.Design.Light"
366     android:visibility="invisible" />
367
368     <android.support.design.widget.FloatingActionButton
369     android:id="@+id/fabEditEventToSearch"
370     android:layout_width="wrap_content"
371     android:layout_height="wrap_content"
372     android:layout_gravity="bottom|end"
373     android:layout_marginBottom="90dp"
374     android:layout_marginEnd="@dimen/fab_margin"
375     android:src="@drawable/ic_search_white_24dp"
376     android:theme="@style/Theme.Design.Light"
377     android:visibility="invisible" />
378
379     <ImageButton
380     android:id="@+id/editEventSaveBtn"
381     android:layout_width="65dp"
382     android:layout_height="65dp"
383     android:layout_marginTop="20dp"
384     android:layout_gravity="bottom|start"
385     android:layout_marginBottom="25dp"
386     android:layout_marginStart="@dimen/fab_margin"
387     android:background="@drawable/ripple_effect"
388     android:contentDescription="@string/profileSaveBtn"
389     android:src="@drawable/ic_save_white_24dp"
390     android:text="@string/profileSaveBtn"
391     android:textColor="#fff"
392     app:layout_constraintEnd_toStartOf="@+id/guideline7"
393     app:layout_constraintHorizontal_bias="0.026"
394     app:layout_constraintStart_toStartOf="@+id/guideline6"
395     app:layout_constraintTop_toBottomOf="@+id/editEventInfo" />
396
397     <ImageButton
398     android:id="@+id/editEventDeleteBtn"
399     android:layout_width="65dp"
400     android:layout_height="65dp"
```

```
401         android:layout_marginTop="20dp"
402         android:layout_gravity="bottom|start"
403         android:layout_marginBottom="25dp"
404         android:layout_marginStart="107dp"
405         android:background="@drawable/ripple_effect"
406         android:contentDescription="@string/profileSaveBtn"
407         android:src="@drawable/ic_delete_forever_white_24dp"
408         android:visibility="invisible"
409         android:text="@string/editEventDeleteBtn"
410         android:textColor="#fff"
411         app:layout_constraintEnd_toStartOf="@+id/guideline7"
412         app:layout_constraintHorizontal_bias="0.026"
413         app:layout_constraintStart_toStartOf="@+id/guideline6"
414         app:layout_constraintTop_toBottomOf="@+id/editEventInfo" />
415
416     <ImageButton
417         android:id="@+id/editEventLeaveBtn"
418         android:layout_width="65dp"
419         android:layout_height="65dp"
420         android:layout_marginTop="20dp"
421         android:layout_gravity="bottom|start"
422         android:layout_marginBottom="25dp"
423         android:layout_marginStart="107dp"
424         android:background="@drawable/ripple_effect"
425         android:contentDescription="@string/profileSaveBtn"
426         android:src="@drawable/ic_exit_to_app_white_24dp"
427         android:visibility="invisible"
428         android:text="@string/editEventLeaveBtn"
429         android:textColor="#fff"
430         app:layout_constraintEnd_toStartOf="@+id/guideline7"
431         app:layout_constraintHorizontal_bias="0.026"
432         app:layout_constraintStart_toStartOf="@+id/guideline6"
433         app:layout_constraintTop_toBottomOf="@+id/editEventInfo" />
434
435     </android.support.design.widget.CoordinatorLayout>
436
437 </RelativeLayout>
438
439 </RelativeLayout>
```

Code-Anhang 18: EditEvent

ListView.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2
3 <CheckedTextView
4     xmlns:android="http://schemas.android.com/apk/res/android"
5     android:id="@+id/checkTextView"
6     android:layout_width="match_parent"
7     android:layout_height="50dp"
8     android:textColor="#000"
9     android:textStyle="bold"
10    android:textSize="17sp"
11    android:checkMarkTint="#000"
12    android:gravity="center_vertical"
13    android:checkMark="?android:attr/listChoiceIndicatorMultiple" />
```

Code-Anhang 19: ListView

Startapp

```
1 public class Startapp extends AppCompatActivity {
2
3     private ImageButton goToLogin;
4     private ImageButton goToRegistration;
5     FirebaseAuth mAuth;
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        requestWindowFeature(Window.FEATURE_NO_TITLE);
11        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
12        ↪ WindowManager.LayoutParams.FLAG_FULLSCREEN);
13        setContentView(R.layout.activity_startapp);
14
15        mAuth = FirebaseAuth.getInstance();
16
17        goToLogin = findViewById(R.id.goToLogin);
18        goToRegistration = findViewById(R.id.goToRegistration);
19
20        goToLogin.setOnClickListener(new View.OnClickListener() {
21            @Override
22            public void onClick(View view) {
23                ImageButton a = (ImageButton) view;
24                a.setEnabled(false);
25
26                Runnable r = new Runnable(){
27                    @Override
28                    public void run(){
29                        long future = System.currentTimeMillis() + 2500;
30                        while(System.currentTimeMillis() < future){
31                            synchronized (this){
32                                try{
33                                    wait(future - System.currentTimeMillis());
34                                }catch(Exception e){
35                                    //...
36                                }
37                            }
38                        }
39                        goToLogin.setEnabled(true);
40                    }
41                };
42                Thread rThread = new Thread(r);
43                rThread.start();
44
45                Intent login = new Intent(Startapp.this, Login.class);
46                startActivity(login);
47            }
48        });
49
50
51
52        goToRegistration.setOnClickListener(new View.OnClickListener() {
53            @Override
54            public void onClick(View view) {
55                ImageButton a = (ImageButton) view;
```



```
56         a.setEnabled(false);
57
58         Runnable r = new Runnable(){
59             @Override
60             public void run(){
61                 long future = System.currentTimeMillis() + 2500;
62                 while(System.currentTimeMillis() < future){
63                     synchronized (this){
64                         try{
65                             wait(future - System.currentTimeMillis());
66                         }catch(Exception e){
67                             //...
68                         }
69                     }
70                 }
71                 goToRegistration.setEnabled(true);
72             }
73         };
74         Thread rThread = new Thread(r);
75         rThread.start();
76
77         Intent register = new Intent(Startapp.this, Register.class);
78         startActivity(register);
79     }
80 });
81
82 }
83
84 @Override
85 protected void onStart() {
86     super.onStart();
87     if(mAuth.getCurrentUser() != null){
88         finish();
89         startActivity(new Intent(this, Mainpage.class));
90     }
91 }
92
93 }
```

Login

```
1 package com.example.tonyk.promac;
2
3 import android.content.Intent;
4 import android.support.annotation.NonNull;
5 import android.support.v7.app.AppCompatActivity;
6 import android.os.Bundle;
7 import android.util.Patterns;
8 import android.view.View;
9 import android.view.Window;
10 import android.view.WindowManager;
11 import android.widget.Button;
12 import android.widget.EditText;
13 import android.widget.ProgressBar;
14 import android.widget.Toast;
15
16 import com.google.android.gms.tasks.OnCompleteListener;
17 import com.google.android.gms.tasks.Task;
18 import com.google.firebase.auth.AuthResult;
```

```
19 import com.google.firebase.auth.FirebaseAuth;
20
21 public class Login extends AppCompatActivity implements View.OnClickListener {
22
23     EditText etEmail;
24     EditText etPass;
25     Button loginBtn;
26     ProgressBar progressBar;
27
28     private FirebaseAuth mAuth;
29
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         requestWindowFeature(Window.FEATURE_NO_TITLE);
34         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
35             ↪ WindowManager.LayoutParams.FLAG_FULLSCREEN);
36         setContentView(R.layout.activity_login);
37
38         mAuth = FirebaseAuth.getInstance();
39
40         etEmail = findViewById(R.id.loginEmail);
41         etPass = findViewById(R.id.loginPass);
42         loginBtn = findViewById(R.id.loginBtn);
43         progressBar = findViewById(R.id.loginProgressBar);
44
45         loginBtn.setOnClickListener(this);
46
47         View dView = getWindow().getDecorView();
48         dView.setSystemUiVisibility(
49             View.SYSTEM_UI_FLAG_LAYOUT_STABLE
50             | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
51             | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
52             | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
53             | View.SYSTEM_UI_FLAG_FULLSCREEN
54             | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY);
55     }
56
57     private void userLogin() {
58         String email = etEmail.getText().toString().trim();
59         String password = etPass.getText().toString().trim();
60
61         if(email.isEmpty()) {
62             etEmail.setError("Email address is required!");
63             etEmail.requestFocus();
64             return;
65         }
66
67         if(!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
68             etEmail.setError("Enter a valid email!");
69             etEmail.requestFocus();
70             return;
71         }
72
73         if(password.isEmpty()) {
74             etPass.setError("Password is required!");
75             etPass.requestFocus();
76             return;
77         }
78
79         if(password.length() < 6) {
```

```
79         etPass.setError("The password should be at least 6 characters!");
80         etPass.requestFocus();
81         return;
82     }
83
84     loginBtn.setEnabled(false);
85     loginBtn.setClickable(false);
86     progressBar.setVisibility(View.VISIBLE);
87     mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(new
88         ↳ OnCompleteListener<AuthResult>() {
89         @Override
90         public void onComplete(@NonNull Task<AuthResult> task) {
91             progressBar.setVisibility(View.GONE);
92             if(task.isSuccessful()){
93                 finish();
94                 Toast.makeText(getApplicationContext(), "Login successful!",
95                     ↳ Toast.LENGTH_SHORT).show();
96                 Intent intent = new Intent(Login.this, Mainpage.class);
97                 intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
98                 startActivity(intent);
99                 loginBtn.setEnabled(true);
100                loginBtn.setClickable(true);
101            }else{
102                //noinspection ConstantConditions
103                Toast.makeText(getApplicationContext(), task.getException().
104                    ↳ getMessage(), Toast.LENGTH_SHORT).show();
105            }
106        }
107    });
108 }
109
110 @Override
111 protected void onStart() {
112     super.onStart();
113     if(mAuth.getCurrentUser() != null){
114         finish();
115         startActivity(new Intent(this, Mainpage.class));
116     }
117 }
118
119 @Override
120 public void onClick(View view) {
121     switch (view.getId()){
122         case R.id.loginBtn:
123             userLogin();
124             break;
125     }
126 }
```

Code-Anhang 20: Anmeldung

Register

```
1 package com.example.tonyk.promac;
2
3 import android.content.Intent;
4 import android.support.annotation.NonNull;
5 import android.support.v7.app.AppCompatActivity;
```

```
6 import android.os.Bundle;
7 import android.util.Patterns;
8 import android.view.View;
9 import android.view.Window;
10 import android.view.WindowManager;
11 import android.widget.Button;
12 import android.widget.EditText;
13 import android.widget.ProgressBar;
14 import android.widget.Toast;
15
16 import com.google.android.gms.tasks.OnCompleteListener;
17 import com.google.android.gms.tasks.Task;
18 import com.google.firebase.auth.AuthResult;
19 import com.google.firebase.auth.FirebaseAuth;
20 import com.google.firebase.auth.FirebaseAuthUserCollisionException;
21
22 public class Register extends AppCompatActivity implements View.OnClickListener {
23
24     private FirebaseAuth mAuth;
25     ProgressBar progressBar;
26     EditText etEmail, etPass, etConfirmPass;
27     Button regBtn;
28
29     @Override
30     protected void onCreate(Bundle savedInstanceState) {
31         super.onCreate(savedInstanceState);
32         requestWindowFeature(Window.FEATURE_NO_TITLE);
33         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
34             ↳ WindowManager.LayoutParams.FLAG_FULLSCREEN);
35         setContentView(R.layout.activity_register);
36
37         mAuth = FirebaseAuth.getInstance();
38
39         etEmail = findViewById(R.id.regEmail);
40         etPass = findViewById(R.id.regPass);
41         etConfirmPass = findViewById(R.id.regConfirmPass);
42         regBtn = findViewById(R.id.regBtn);
43         progressBar = findViewById(R.id.regProgressBar);
44
45         regBtn.setOnClickListener(this);
46
47         View dView = getWindow().getDecorView();
48         dView.setSystemUiVisibility(
49             View.SYSTEM_UI_FLAG_LAYOUT_STABLE
50             | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
51             | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
52             | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
53             | View.SYSTEM_UI_FLAG_FULLSCREEN
54             | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY);
55     }
56
57     private void registerUser() {
58         String email = etEmail.getText().toString().trim();
59         String password = etPass.getText().toString().trim();
60         String confirmPass = etConfirmPass.getText().toString().trim();
61
62         if(email.isEmpty()) {
63             etEmail.setError("Email address is required!");
64             etEmail.requestFocus();
65             return;
66         }
67     }
68 }
```

```
66
67     if(!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
68         etEmail.setError("Enter a valid email!");
69         etEmail.requestFocus();
70         return;
71     }
72
73     if(password.isEmpty()){
74         etPass.setError("Password is required!");
75         etPass.requestFocus();
76         return;
77     }
78
79     if(password.length() < 6){
80         etPass.setError("The password should be at least 6 characters!");
81         etPass.requestFocus();
82         return;
83     }
84
85     if(password.equals(confirmPass)){
86         regBtn.setClickable(false);
87         regBtn.setEnabled(false);
88         progressBar.setVisibility(View.VISIBLE);
89         mAuth.createUserWithEmailAndPassword(email, password).
90             ↪ addOnCompleteListener(new OnCompleteListener<AuthResult>() {
91             @Override
92             public void onComplete(@NonNull Task<AuthResult> task) {
93                 progressBar.setVisibility(View.GONE);
94                 if (task.isSuccessful()) {
95                     finish();
96                     Toast.makeText(getApplicationContext(), "Your registration
97                         ↪ was successful!", Toast.LENGTH_SHORT).show();
98                     startActivity(new Intent(Register.this, Mainpage.class));
99                     regBtn.setClickable(true);
100                     regBtn.setEnabled(true);
101                 }else{
102                     if(task.getException() instanceof
103                         ↪ FirebaseAuthUserCollisionException){
104                         Toast.makeText(getApplicationContext(), "Email is
105                             ↪ already taken!", Toast.LENGTH_SHORT).show();
106                         regBtn.setClickable(true);
107                         regBtn.setEnabled(true);
108                     }else {
109                         Toast.makeText(getApplicationContext(), "Email is not
110                             ↪ valid!", Toast.LENGTH_SHORT).show();
111                         regBtn.setClickable(true);
112                         regBtn.setEnabled(true);
113                     }
114                 }
115             }
116             });
117     }else{
118         etPass.setError("The passwords do not match!");
119         etPass.requestFocus();
120     }
121 }
122
123 @Override
124 public void onClick(View view) {
125     switch(view.getId()){
126         case R.id.regBtn:
```

```
122         registerUser();
123         break;
124     }
125 }
126 }
```

Code-Anhang 21: Registrierung

Mainpage

```
1 package com.example.tonyk.promac;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.support.v7.widget.CardView;
7 import android.view.View;
8 import android.view.Window;
9 import android.view.WindowManager;
10 import android.widget.GridLayout;
11 import android.widget.ImageButton;
12
13 import com.google.firebase.auth.FirebaseAuth;
14 import com.google.firebase.auth.FirebaseUser;
15 import com.google.firebase.database.DatabaseReference;
16 import com.google.firebase.database.FirebaseDatabase;
17
18 public class Mainpage extends AppCompatActivity {
19
20     GridLayout mainGrid;
21     ImageButton logoutBtn;
22     FirebaseAuth mAuth;
23
24     @Override
25     protected void onCreate(Bundle savedInstanceState) {
26         super.onCreate(savedInstanceState);
27         requestWindowFeature(Window.FEATURE_NO_TITLE);
28         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
29             ↪ WindowManager.LayoutParams.FLAG_FULLSCREEN);
30         setContentView(R.layout.activity_mainpage);
31
32         mAuth = FirebaseAuth.getInstance();
33         FirebaseDatabase mDatabase = FirebaseDatabase.getInstance();
34         DatabaseReference mRef = mDatabase.getReference("Benutzer/");
35
36         FirebaseUser user = mAuth.getCurrentUser();
37         if (user != null) {
38             String email = user.getEmail();
39             String userID = user.getId();
40             mRef.child(userID).child("Email").setValue(email);
41             mRef.child(userID).child("UserID").setValue(userID);
42         }
43
44         mainGrid = findViewById(R.id.mainGrid);
45         setSingleEvent(mainGrid);
46         logoutBtn = findViewById(R.id.logoutBtn);
47
48         logoutBtn.setOnClickListener(new View.OnClickListener() {
49             @Override
```

```
49         public void onClick(View view) {
50             FirebaseAuth.getInstance().signOut();
51             finish();
52             startActivity(new Intent(Mainpage.this, Startapp.class));
53         }
54     });
55
56     View dView = getWindow().getDecorView();
57     dView.setSystemUiVisibility(
58         View.SYSTEM_UI_FLAG_LAYOUT_STABLE
59         | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
60         | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
61         | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
62         | View.SYSTEM_UI_FLAG_FULLSCREEN
63         | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY);
64     }
65
66     private void setSingleEvent(GridLayout mainGrid){
67         for(int i = 0; i < mainGrid.getChildCount(); i++){
68             CardView cV = (CardView)mainGrid.getChildAt(i);
69             final int var = i;
70             cV.setOnClickListener(new View.OnClickListener() {
71                 @Override
72                 public void onClick(View view) {
73                     if(var == 0){
74                         Intent i = new Intent(Mainpage.this, Profile.class);
75                         startActivity(i);
76                     }else //noinspection StatementWithEmptyBody
77                     if(var == 1){
78                         //Intent i = new Intent(Mainpage.this, Messages.class);
79                         //startActivity(i);
80                     }else if(var == 2){
81                         Intent i = new Intent (Mainpage.this, InviteMembers.class);
82                         startActivity(i);
83                     }else if(var == 3){
84                         Intent i = new Intent(Mainpage.this, Search.class);
85                         startActivity(i);
86                     }else if(var == 4){
87                         Intent i = new Intent (Mainpage.this, Events.class);
88                         startActivity(i);
89                     }else //noinspection StatementWithEmptyBody
90                     if(var == 5){
91                         //Intent i = new Intent(Mainpage.this, Calender.class);
92                         //startActivity(i);
93                     }
94                 }
95             });
96         }
97     }
98 }
```

Code-Anhang 22: Hauptseite

Wichtig: Alle Klassen nach Profile.java sind reduziert und haben keine Imports, FAB Buttons mit Animationen und refreshUI-Funktion mit onResume, um die Doku möglichst kleiner zu machen.

Profile

```
1 package com.example.tonyk.promac;
2
3 import android.content.Intent;
4 import android.graphics.Bitmap;
5 import android.net.Uri;
6 import android.provider.MediaStore;
7 import android.support.annotation.NonNull;
8 import android.support.v7.app.AppCompatActivity;
9 import android.os.Bundle;
10 import android.support.design.widget.FloatingActionButton;
11 import android.view.Menu;
12 import android.view.MenuItem;
13 import android.view.View;
14 import android.view.Window;
15 import android.view.WindowManager;
16 import android.view.animation.Animation;
17 import android.view.animation.AnimationUtils;
18 import android.widget.EditText;
19 import android.widget.ImageButton;
20 import android.widget.ImageView;
21 import android.widget.ProgressBar;
22 import android.widget.TextView;
23 import android.widget.Toast;
24
25 import com.bumptech.glide.Glide;
26 import com.google.android.gms.tasks.OnCompleteListener;
27 import com.google.android.gms.tasks.OnFailureListener;
28 import com.google.android.gms.tasks.OnSuccessListener;
29 import com.google.android.gms.tasks.Task;
30 import com.google.firebase.auth.FirebaseAuth;
31 import com.google.firebase.auth.FirebaseUser;
32 import com.google.firebase.auth.UserProfileChangeRequest;
33 import com.google.firebase.database.DataSnapshot;
34 import com.google.firebase.database.DatabaseError;
35 import com.google.firebase.database.DatabaseReference;
36 import com.google.firebase.database.FirebaseDatabase;
37 import com.google.firebase.database.ValueEventListener;
38 import com.google.firebase.storage.FirebaseStorage;
39 import com.google.firebase.storage.StorageReference;
40 import com.google.firebase.storage.UploadTask;
41
42 import java.io.IOException;
43
44 public class Profile extends AppCompatActivity {
45
46     private static final int imageNr = 24;
47     Uri uriProfileImage;
48     String imageUrl;
49     FirebaseAuth mAuth;
50     final FirebaseDatabase mDatabase = FirebaseDatabase.getInstance();
51     DatabaseReference mRef = mDatabase.getReference("Benutzer/");
52
53     FloatingActionButton fabOpenProfile, fabProfileToCreateEvent,
54         fabProfileToEditEvent, fabProfileToSearch, fabProfileToMenu;
55     Animation fab_open, fab_close, rotate_btn_forward, rotate_btn_backward;
56     boolean isOpen = false;
57
58     ProgressBar progressBar;
59     ImageView profileInfoPhoto;
60     TextView profileInfoEmail, uploadPhoto;
61     EditText profileInfoName, profileInfoTelephoneText, profileInfoLeitzeichenText,
```



```

62         profileInfoDepartmentText, profileInfoRoomText,
           ↪ profileInfoAdditionalText;
63     ImageButton profileSaveBtn, uploadBtn;
64
65
66     @Override
67     protected void onCreate(final Bundle savedInstanceState) {
68         super.onCreate(savedInstanceState);
69         refreshUI();
70         requestWindowFeature(Window.FEATURE_NO_TITLE);
71         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
           ↪ WindowManager.LayoutParams.FLAG_FULLSCREEN);
72         getWindow().setSoftInputMode(WindowManager.LayoutParams.
           ↪ SOFT_INPUT_STATE_HIDDEN);
73         setContentView(R.layout.activity_profile);
74
75         mAuth = FirebaseAuth.getInstance();
76
77         //Other objects
78         progressBar = findViewById(R.id.profileProgressBar);
79         profileInfoPhoto = findViewById(R.id.userInfoPhoto);
80         profileInfoEmail = findViewById(R.id.userInfoEmail);
81         profileInfoName = findViewById(R.id.userInfoName);
82         profileInfoTelephoneText = findViewById(R.id.userInfoTelephoneText);
83         profileInfoLeitzeichenText = findViewById(R.id.userInfoLeitzeichenText);
84         profileInfoDepartmentText = findViewById(R.id.userInfoDepartmentText);
85         profileInfoRoomText = findViewById(R.id.userInfoRoomText);
86         profileInfoAdditionalText = findViewById(R.id.userInfoAdditionalText);
87         profileSaveBtn = findViewById(R.id.profileSaveBtn);
88         uploadPhoto = findViewById(R.id.uploadPhoto);
89         uploadBtn = findViewById(R.id.uploadBtn);
90
91         //FAB Buttons
92         fabOpenProfile = findViewById(R.id.fabOpenProfile);
93         fabProfileToCreateEvent = findViewById(R.id.fabProfileToCreateEvent);
94         fabProfileToEditEvent = findViewById(R.id.fabProfileToEditEvent);
95         fabProfileToSearch = findViewById(R.id.fabProfileToSearch);
96         fabProfileToMenu = findViewById(R.id.fabProfileToMenu);
97
98         //Animations
99         fab_open = AnimationUtils.loadAnimation(this, R.anim.fab_open);
100        fab_close = AnimationUtils.loadAnimation(this, R.anim.fab_close);
101        rotate_btn_backward = AnimationUtils.loadAnimation(this, R.anim.
           ↪ rotate_btn_backward);
102        rotate_btn_forward = AnimationUtils.loadAnimation(this, R.anim.
           ↪ rotate_btn_forward);
103
104        loadUserInfo();
105        loadUserPhoto();
106
107        fabOpenProfile.setOnClickListener(new View.OnClickListener() {
108            @Override
109            public void onClick(View view) {
110                animateFab();
111            }
112        });
113
114        fabProfileToCreateEvent.setOnClickListener(new View.OnClickListener() {
115            @Override
116            public void onClick(View view) {
117                Intent i = new Intent(Profile.this, CreateEvent.class);

```

```
118         startActivity(i);
119         animateFab();
120     }
121 });
122
123 fabProfileToEditEvent.setOnClickListener(new View.OnClickListener() {
124     @Override
125     public void onClick(View view) {
126         Intent i = new Intent(Profile.this, Events.class);
127         startActivity(i);
128         animateFab();
129     }
130 });
131
132 fabProfileToSearch.setOnClickListener(new View.OnClickListener() {
133     @Override
134     public void onClick(View view) {
135         Intent i = new Intent(Profile.this, Search.class);
136         startActivity(i);
137         animateFab();
138     }
139 });
140
141
142 fabProfileToMenu.setOnClickListener(new View.OnClickListener() {
143     @Override
144     public void onClick(View view) {
145         Intent i = new Intent(Profile.this, Mainpage.class);
146         startActivity(i);
147         animateFab();
148     }
149 });
150
151 profileInfoPhoto.setOnClickListener(new View.OnClickListener() {
152     @Override
153     public void onClick(View view) {
154         chooseImage();
155     }
156 });
157
158 uploadBtn.setOnClickListener(new View.OnClickListener() {
159     @Override
160     public void onClick(View view) {
161         if (saveUserPhoto()) {
162             saveUserPhoto();
163             uploadBtn.setEnabled(false);
164             uploadPhoto.setVisibility(View.GONE);
165             uploadBtn.setVisibility(View.GONE);
166             profileSaveBtn.setVisibility(View.VISIBLE);
167             profileInfoPhoto.setEnabled(false);
168             profileInfoPhoto.setClickable(false);
169         }
170     }
171 });
172
173 profileSaveBtn.setOnClickListener(new View.OnClickListener() {
174     @Override
175     public void onClick(View view) {
176         if (saveUserInfo()) {
177             saveUserInfo();
178             Toast.makeText(Profile.this, "Info was updated", Toast.
```

```

179         ↪ LENGTH_SHORT).show();
180         Intent i = new Intent(Profile.this, Mainpage.class);
181         startActivity(i);
182         return;
183     }
184     Toast.makeText(Profile.this, "Nothing was updated", Toast.
185         ↪ LENGTH_SHORT).show();
186     Intent i = new Intent(Profile.this, Mainpage.class);
187     startActivity(i);
188 }
189
190 public void refreshUI() {
191     final View v = getWindow().getDecorView();
192     v.setOnSystemUiVisibilityChangeListener(new View.
193         ↪ OnSystemUiVisibilityChangeListener() {
194         @Override
195         public void onSystemUiVisibilityChange(int visibility) {
196             if ((visibility & View.SYSTEM_UI_FLAG_FULLSCREEN) == 0) {
197                 v.setSystemUiVisibility(
198                     View.SYSTEM_UI_FLAG_LAYOUT_STABLE
199                     | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
200                     | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
201                     | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
202                     | View.SYSTEM_UI_FLAG_FULLSCREEN
203                     | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY);
204             }
205         });
206     }
207
208     private boolean saveUserInfo() {
209         FirebaseUser user = mAuth.getCurrentUser();
210         if (user != null) {
211             String userID = user.getId();
212             String name = profileInfoName.getText().toString();
213             mRef.child(userID).child("Name").setValue(name);
214             String leitzeichen = profileInfoLeitzeichenText.getText().toString();
215             mRef.child(userID).child("Leitzeichen").setValue(leitzeichen);
216             String department = profileInfoDepartmentText.getText().toString();
217             mRef.child(userID).child("Abteilung").setValue(department);
218             String room = profileInfoRoomText.getText().toString();
219             mRef.child(userID).child("Raum").setValue(room);
220             String phone = profileInfoTelephoneText.getText().toString();
221             mRef.child(userID).child("Telefon").setValue(phone);
222             String info = profileInfoAdditionalText.getText().toString();
223             mRef.child(userID).child("Additional Info").setValue(info);
224
225             return true;
226         }
227         return false;
228     }
229
230     private void loadUserInfo() {
231         FirebaseUser user = mAuth.getCurrentUser();
232         if (user != null) {
233             String userID = user.getId();
234             DatabaseReference mUserRef = mRef.child(userID);
235             mUserRef.addValueEventListener(new ValueEventListener() {
236                 @Override

```

```
237         public void onDataChange(DataSnapshot dataSnapshot) {
238             String name = dataSnapshot.child("Name").getValue(String.class);
239             profileInfoName.setText(name);
240             String leitzeichen = dataSnapshot.child("Leitzeichen").getValue(
241                 ↪ String.class);
242             profileInfoLeitzeichenText.setText(leitzeichen);
243             String room = dataSnapshot.child("Raum").getValue(String.class);
244             profileInfoRoomText.setText(room);
245             String tel = dataSnapshot.child("Telefon").getValue(String.class
246                 ↪ );
247             profileInfoTelephoneText.setText(tel);
248             String department = dataSnapshot.child("Abteilung").getValue(
249                 ↪ String.class);
250             profileInfoDepartmentText.setText(department);
251             String info = dataSnapshot.child("Additional Info").getValue(
252                 ↪ String.class);
253             profileInfoAdditionalText.setText(info);
254         }
255
256         @Override
257         public void onCancelled(DatabaseError databaseError) {
258
259         }
260     };
261
262     @Override
263     protected void onStart() {
264         super.onStart();
265         FirebaseUser user = mAuth.getCurrentUser();
266         if (user != null) {
267             profileInfoEmail.setText(mAuth.getCurrentUser().getEmail());
268             if (mAuth.getCurrentUser() == null) {
269                 finish();
270                 startActivity(new Intent(Profile.this, Startapp.class));
271             }
272         }
273
274     @Override
275     public void onResume() {
276         super.onResume();
277         refreshUI();
278     }
279
280     private void loadUserPhoto() {
281         FirebaseUser user = mAuth.getCurrentUser();
282         if (user != null) {
283             if (user.getPhotoUrl() != null) {
284                 Glide.with(this)
285                     .load(user.getPhotoUrl().toString())
286                     .into(profileInfoPhoto);
287             }
288         }
289
290     private boolean saveUserPhoto() {
291         FirebaseUser user = mAuth.getCurrentUser();
292         if (user != null) {
293             UserProfileChangeRequest profile = new UserProfileChangeRequest.Builder
```

```

    ↪ ()
294         .setPhotoUri(Uri.parse(imageUrl))
295         .build();
296
297     progressBar.setVisibility(View.VISIBLE);
298     getWindow().setFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE,
299         WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
300
301     user.updateProfile(profile)
302         .addOnCompleteListener(new OnCompleteListener<Void>() {
303             @Override
304             public void onComplete(@NonNull Task<Void> task) {
305                 if (task.isSuccessful()) {
306                     progressBar.setVisibility(View.GONE);
307                     getWindow().clearFlags(WindowManager.LayoutParams.
308                         ↪ FLAG_NOT_TOUCHABLE);
309                     Toast.makeText(Profile.this, "Photo was uploaded",
310                         ↪ Toast.LENGTH_SHORT).show();
311                 }
312             }
313         });
314     return true;
315 }
316
317 private void chooseImage() {
318     Intent i = new Intent();
319     i.setType("image/*");
320     i.setAction(Intent.ACTION_GET_CONTENT);
321     startActivityForResult(Intent.createChooser(i, "Image is selected"), imageNr
322         ↪ );
323 }
324
325 @Override
326 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
327     super.onActivityResult(requestCode, resultCode, data);
328     if (requestCode == imageNr && resultCode == RESULT_OK && data != null &&
329         ↪ data.getData() != null) {
330         uriProfileImage = data.getData();
331         try {
332             Bitmap bm = MediaStore.Images.Media.getBitmap(getContentResolver(),
333                 ↪ uriProfileImage);
334             profileInfoPhoto.setImageBitmap(bm);
335             uploadImage();
336         } catch (IOException e) {
337             e.printStackTrace();
338         }
339     }
340 }
341
342 private void uploadImage() {
343     StorageReference imageRef = FirebaseStorage.getInstance()
344         .getReference("profilePics/" + System.currentTimeMillis() + ".jpg");
345     if (uriProfileImage != null) {
346         imageRef.putFile(uriProfileImage).addOnSuccessListener(new
347             ↪ OnSuccessListener<UploadTask.TaskSnapshot>() {
348                 @Override
349                 public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
350                     //noinspection ConstantConditions
351                     imageUrl = taskSnapshot.getDownloadUrl().toString();
352                 }
353             });
354     }
355 }
```

```
348         uploadPhoto.setVisibility(View.VISIBLE);
349         uploadBtn.setVisibility(View.VISIBLE);
350         profileSaveBtn.setVisibility(View.INVISIBLE);
351     }
352     }).addOnFailureListener(new OnFailureListener() {
353         @Override
354         public void onFailure(@NonNull Exception e) {
355             Toast.makeText(Profile.this, e.getMessage(), Toast.LENGTH_SHORT)
356                 ↪ .show();
357         }
358     });
359 }
360
361 private void animateFab() {
362     if (isOpen) {
363         fabOpenProfile.startAnimation(rotate_btn_forward);
364         fabProfileToCreateEvent.startAnimation(fab_close);
365         fabProfileToEditEvent.startAnimation(fab_close);
366         fabProfileToSearch.startAnimation(fab_close);
367         fabProfileToMenu.startAnimation(fab_close);
368         fabProfileToCreateEvent.setClickable(false);
369         fabProfileToEditEvent.setClickable(false);
370         fabProfileToSearch.setClickable(false);
371         fabProfileToMenu.setClickable(false);
372         isOpen = false;
373     } else {
374         fabOpenProfile.startAnimation(rotate_btn_backward);
375         fabProfileToCreateEvent.startAnimation(fab_open);
376         fabProfileToEditEvent.startAnimation(fab_open);
377         fabProfileToSearch.startAnimation(fab_open);
378         fabProfileToMenu.startAnimation(fab_open);
379         fabProfileToCreateEvent.setClickable(true);
380         fabProfileToEditEvent.setClickable(true);
381         fabProfileToSearch.setClickable(true);
382         fabProfileToMenu.setClickable(true);
383         isOpen = true;
384     }
385 }
386
387 @Override
388 public boolean onCreateOptionsMenu(Menu menu) {
389     getMenuInflater().inflate(R.menu.menu_main, menu);
390     return true;
391 }
392
393 @Override
394 public boolean onOptionsItemSelected(MenuItem item) {
395     int id = item.getItemId();
396     return (id == R.id.action_settings) || super.onOptionsItemSelected(item);
397 }
398 }
```

Code-Anhang 23: Profileseite

InviteMembers

```
1 public class InviteMembers extends AppCompatActivity implements AdapterView.
    ↪ OnItemClickListener {
```

```
2
3 FirebaseDatabase mDatabase = FirebaseDatabase.getInstance();
4 DatabaseReference mRefB = mDatabase.getReference("Benutzer/");
5
6 private boolean exec = true;
7 private String members = "";
8 ImageButton inviteMembersBtn;
9 ListView inviteMembersListView;
10 SearchView inviteMembersSearch;
11
12 private ArrayAdapter<String> adapter;
13 FirebaseAuth mAuth;
14
15 private ArrayList<String> selectedUsers = new ArrayList<>();
16 private ArrayList<String> allUsers = new ArrayList<>();
17
18 @Override
19 protected void onCreate(Bundle savedInstanceState) {
20     super.onCreate(savedInstanceState);
21     requestWindowFeature(Window.FEATURE_NO_TITLE);
22     getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
23         ↪ WindowManager.LayoutParams.FLAG_FULLSCREEN);
24     getWindow().setSoftInputMode(WindowManager.LayoutParams.
25         ↪ SOFT_INPUT_STATE_HIDDEN);
26     setContentView(R.layout.activity_invite_members);
27
28     adapter = new ArrayAdapter<>(this, R.layout.checkbox_listview, R.id.
29         ↪ checkTextView, allUsers);
30
31     mAuth = FirebaseAuth.getInstance();
32     final FirebaseUser user = mAuth.getCurrentUser();
33     if (user != null) {
34         mRefB.addListenerForSingleValueEvent(new ValueEventListener() {
35             @Override
36             public void onDataChange(DataSnapshot dataSnapshot) {
37                 for (DataSnapshot ds : dataSnapshot.getChildren()) {
38                     String email = ds.child("Email").getValue(String.class);
39                     String uid = ds.child("UserID").getValue(String.class);
40                     assert uid != null;
41                     if (!uid.equals(user.getUid()))
42                         allUsers.add(email);
43                 }
44             }
45             @Override
46             public void onCancelled(DatabaseError databaseError) {
47             }
48         });
49     }
50     inviteMembersListView.setChoiceMode(AbsListView.CHOICE_MODE_MULTIPLE);
51     inviteMembersListView.setAdapter(adapter);
52     inviteMembersListView.setOnItemClickListener(new AdapterView.
53         ↪ OnItemClickListener() {
54         @Override
55         public void onItemClick(AdapterView<?> adapterView, View view, int i,
56             ↪ long l) {
57             String selectedUser = ((TextView) view).getText().toString();
58             if (selectedUsers.contains(selectedUser)) {
59                 selectedUsers.remove(selectedUser);
60             } else {
```

```
58         selectedUsers.add(selectedUser);
59     }
60 }
61 });
62
63 inviteMembersSearch.setOnQueryTextListener(new SearchView.
64     ↪ OnQueryTextListener() {
65     @Override
66     public boolean onQueryTextSubmit(String text) {
67         return false;
68     }
69
70     @Override
71     public boolean onQueryTextChange(String newText) {
72         adapter.getFilter().filter(newText);
73         return false;
74     }
75 });
76
77 inviteMembersSearch.setOnClickListener(new View.OnClickListener() {
78     @Override
79     public void onClick(View view) {
80         inviteMembersSearch.setIconified(false);
81     }
82 });
83
84 inviteMembersBtn.setOnClickListener(new View.OnClickListener() {
85     @Override
86     public void onClick(View view) {
87         invitedUsers();
88         if (exec) {
89             Intent i = new Intent(InviteMembers.this, CreateEvent.class);
90             i.putExtra("members", members);
91             i.putExtra("selectedUsers", selectedUsers);
92             startActivity(i);
93         }
94     });
95
96 }
97
98 private void invitedUsers() {
99     if (!selectedUsers.isEmpty()) {
100         for (String user : selectedUsers) {
101             //noinspection StringConcatenationInLoop
102             members += user + ", ";
103         }
104         members = members.substring(0, members.length() - 2);
105     } else {
106         Toast.makeText(getApplicationContext(), "Nobody invited!", Toast.
107             ↪ LENGTH_SHORT).show();
108         Intent i = new Intent(InviteMembers.this, Mainpage.class);
109         startActivity(i);
110         exec = false;
111     }
112 }
113
114 @Override
115 public boolean onCreateOptionsMenu(Menu menu) {
116     getMenuInflater().inflate(R.menu.menu_main, menu);
117     return true;
118 }
```



```
117     }
118
119     @Override
120     public boolean onOptionsItemSelected(MenuItem item) {
121         int id = item.getItemId();
122         return (id == R.id.action_settings) || super.onOptionsItemSelected(item);
123     }
124
125     @Override
126     public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
127         //...
128     }
129
130
131 }
```

Code-Anhang 24: Teilnehmer einladen

CreateEvent

```
1 public class CreateEvent extends AppCompatActivity {
2
3     FirebaseAuth mAuth;
4     FirebaseDatabase mDatabase = FirebaseDatabase.getInstance();
5     DatabaseReference mRefE = mDatabase.getReference("Events/");
6     DatabaseReference mRefB = mDatabase.getReference("Benutzer/");
7     DatabaseReference mRefBE = mDatabase.getReference("Benutzer-Events/");
8
9     String eventKey, members;
10    ArrayList<String> selectedUsers;
11
12    CoordinatorLayout createEventLayout;
13    TextView createEventMembers;
14    EditText createEventDeadline, createEventName, createEventPurposeText,
15        ↪ createEventPlanText,
16        createEventPlaceText, createEventPriceText, createEventInfoText;
17    ImageButton createEventInviteMembersBtn, createEventSaveBtn;
18
19    @Override
20    protected void onCreate(Bundle savedInstanceState) {
21        super.onCreate(savedInstanceState);
22        requestWindowFeature(Window.FEATURE_NO_TITLE);
23        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
24            ↪ WindowManager.LayoutParams.FLAG_FULLSCREEN);
25        getWindow().setSoftInputMode(WindowManager.LayoutParams.
26            ↪ SOFT_INPUT_STATE_HIDDEN);
27        setContentView(R.layout.activity_create_event);
28
29        mAuth = FirebaseAuth.getInstance();
30
31        createEventLayout = findViewById(R.id.createEventLayout);
32        createEventName = findViewById(R.id.createEventName);
33        createEventMembers = findViewById(R.id.createEventMembers);
34        createEventDeadline = findViewById(R.id.createEventDeadline);
35        createEventPurposeText = findViewById(R.id.createEventPurposeText);
36        createEventPlanText = findViewById(R.id.createEventPlanText);
37        createEventPlaceText = findViewById(R.id.createEventPlaceText);
38        createEventPriceText = findViewById(R.id.createEventPriceText);
39        createEventInfoText = findViewById(R.id.createEventInfoText);
```

```
37         createEventInviteMembersBtn = findViewById(R.id.createEventInviteMembersBtn)
38         ↪ ;
39         createEventSaveBtn = findViewById(R.id.createEventSaveBtn);
40         Bundle bundle = getIntent().getExtras();
41         if (bundle != null) {
42             members = bundle.getString("members");
43         }
44
45         if (bundle != null) {
46             selectedUsers = bundle.getStringArrayList("selectedUsers");
47         }
48
49         createEventMembers.setText(members);
50         createEventMembers.setMovementMethod(new ScrollingMovementMethod());
51
52     }
53
54     private void performClick() {
55         createEventLayout.setOnClickListener(new View.OnClickListener() {
56             @Override
57             public void onClick(View view) {
58
59             }
60         });
61     }
62
63     private void createEvent() {
64         final FirebaseAuth user = mAuth.getCurrentUser();
65         final DatabaseReference mRefDB = mRefE.push();
66
67         if (user != null) {
68             final String eventName = createEventName.getText().toString();
69             if (eventName.trim().isEmpty()) {
70                 createEventName.setError("Name field is empty!");
71                 createEventName.requestFocus();
72                 return;
73             }
74
75             mRefE.addListenerForSingleValueEvent(new ValueEventListener() {
76                 @Override
77                 public void onDataChange(DataSnapshot dataSnapshot) {
78                     for (DataSnapshot ds : dataSnapshot.getChildren()) {
79                         String valueEditText = createEventName.getText().toString();
80                         ↪ //same as eventName, just for better understanding of
81                         ↪ the code
82                         String valueDB = ds.child("Name").getValue(String.class);
83                         assert valueDB != null;
84                         if (valueDB.equals(valueEditText)) {
85                             createEventName.setError("Name is taken!");
86                             createEventName.requestFocus();
87                             return;
88                         }
89                     }
90
91                     mRefDB.child("Name").setValue(eventName);
92                     String eventCreator = user.getEmail();
93                     mRefDB.child("Creator").setValue(eventCreator);
94
95                     String eventDeadline = createEventDeadline.getText().toString();
96                     mRefDB.child("Deadline").setValue(eventDeadline);
97                 }
98             });
99         }
100     }
```

```
95
96         mRefBE.addListenerForSingleValueEvent(new ValueEventListener() {
97             @Override
98             public void onDataChange(DataSnapshot dataSnapshot) {
99                 final String UID = user.getUid();
100                 mRefBE.child(eventKey).child(UID).setValue("Admin");
101                 finish();
102             }
103
104             @Override
105             public void onCancelled(DatabaseError databaseError) {
106
107             }
108         });
109
110         mRefDB.child("ListMembers").setValue(selectedUsers);
111         String eventMembers = createEventMembers.getText().toString();
112         mRefDB.child("Members").setValue(eventMembers);
113
114         String eventPurpose = createEventPurposeText.getText().toString
115             ↪ ();
116         mRefDB.child("Purpose").setValue(eventPurpose);
117         String eventPlan = createEventPlanText.getText().toString();
118         mRefDB.child("Plan").setValue(eventPlan);
119         String eventPlace = createEventPlaceText.getText().toString();
120         mRefDB.child("Place").setValue(eventPlace);
121         String eventPrice = createEventPriceText.getText().toString();
122         mRefDB.child("Price").setValue(eventPrice);
123         String eventInfo = createEventInfoText.getText().toString();
124         mRefDB.child("Information").setValue(eventInfo);
125         eventKey = mRefDB.getKey();
126         mRefDB.child("EventID").setValue(eventKey);
127
128         mRefB.addValueEventListener(new ValueEventListener() {
129             @Override
130             public void onDataChange(DataSnapshot dataSnapshot) {
131                 for(DataSnapshot ds : dataSnapshot.getChildren()){
132                     String uid = ds.child("UserID").getValue(String.
133                         ↪ class);
134                     assert uid != null;
135                     String email = ds.child("Email").getValue(String.
136                         ↪ class);
137                     assert email != null;
138                     for(String emailInvite : selectedUsers){
139                         if(emailInvite.equals(email)){
140                             mRefBE.child(eventKey).child(uid).setValue(
141                                 ↪ email);
142                         }
143                     }
144                 }
145             }
146
147             @Override
148             public void onCancelled(DatabaseError databaseError) {
149
150             }
151         });
152
153         Toast.makeText(CreateEvent.this, "Event was created", Toast.
154             ↪ LENGTH_SHORT).show();
155         Intent i = new Intent(CreateEvent.this, Mainpage.class);
```

```
151         startActivity(i);
152     }
153
154     @Override
155     public void onCancelled(DatabaseError databaseError) {
156
157     }
158     });
159 }
160 }
161
162 @Override
163 protected void onStart() {
164     super.onStart();
165     createEventDeadline.setOnFocusChangeListener(new View.OnFocusChangeListener
166         ↪ () {
167         @Override
168         public void onFocusChange(View view, boolean hasFocus) {
169             if (hasFocus) {
170                 DateDialog dialog = new DateDialog(view);
171                 FragmentTransaction ft = getFragmentManager().beginTransaction()
172                 ↪ ;
173                 dialog.show(ft, "Choose a deadline");
174             }
175         });
176     }
177
178     @Override
179     public boolean onCreateOptionsMenu(Menu menu) {
180         getMenuInflater().inflate(R.menu.menu_main, menu);
181         return true;
182     }
183
184     @Override
185     public boolean onOptionsItemSelected(MenuItem item) {
186         int id = item.getItemId();
187         return (id == R.id.action_settings) || super.onOptionsItemSelected(item);
188     }
189 }
```

Code-Anhang 25: Event erstellen

Events

```
1 public class Events extends AppCompatActivity {
2
3     FirebaseDatabase mDatabase = FirebaseDatabase.getInstance();
4     DatabaseReference mRef = mDatabase.getReference("Events/");
5     FirebaseAuth mAuth;
6
7     String text;
8
9     ListView listView;
10    SearchView eventsSearch;
11
12    ArrayList<String> list = new ArrayList<>();
13    ArrayAdapter<String> adapter;
14 }
```

```
15  @Override
16  protected void onCreate(Bundle savedInstanceState) {
17      super.onCreate(savedInstanceState);
18      requestWindowFeature(Window.FEATURE_NO_TITLE);
19      getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
20          ↪ WindowManager.LayoutParams.FLAG_FULLSCREEN);
21      getWindow().setSoftInputMode(WindowManager.LayoutParams.
22          ↪ SOFT_INPUT_STATE_HIDDEN);
23      setContentView(R.layout.activity_events);
24
25      mAuth = FirebaseAuth.getInstance();
26
27      eventsSearch = findViewById(R.id.eventsSearch);
28      listView = findViewById(R.id.listView);
29
30      adapter = new ArrayAdapter<>(this, android.R.layout.
31          ↪ simple_list_item_activated_1, list);
32      listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
33      listView.setItemChecked(1, true);
34      listView.setAdapter(adapter);
35
36      listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
37          @Override
38          public void onItemClick(AdapterView<?> adapterView, View view, int i,
39              ↪ long l) {
40              for (int nr = 0; nr < list.size(); nr++) {
41                  text = listView.getItemAtPosition(i).toString();
42                  if (nr == i) {
43                      Intent intent = new Intent(Events.this, EditEvent.class);
44                      intent.putExtra("Name", text);
45                      intent.putExtra("listEvents", list);
46                      startActivityForResult(intent, 24);
47                      break;
48                  }
49              }
50          });
51
52      eventsSearch.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
53          @Override
54          public boolean onQueryTextSubmit(String text) {
55              return false;
56          }
57
58          @Override
59          public boolean onQueryTextChange(String newText) {
60              adapter.getFilter().filter(newText);
61              return false;
62          }
63      });
64
65      mRef.addValueEventListener(new ValueEventListener() {
66          @Override
67          public void onDataChange(DataSnapshot dataSnapshot) {
68              for (final DataSnapshot ds : dataSnapshot.getChildren()) {
69                  final String nameID = ds.child("Name").getValue(String.class);
70                  final String creator = ds.child("Creator").getValue(String.class)
71                      ↪ );
72                  ArrayList al = (ArrayList) ds.child("ListMembers").getValue();
73                  assert al != null;
```

```

71         for(Object member : al) {
72             FirebaseAuth user = mAuth.getCurrentUser();
73             if (user != null) {
74                 Object memb = member.toString();
75                 assert creator != null;
76                 if (ds.hasChildren() && !list.contains(nameID) && ((memb
                    ↪ .equals(user.getEmail())) || (creator.equals(user
                    ↪ .getEmail())))) {
77                     list.add(nameID);
78                     adapter.notifyDataSetChanged();
79                 }
80             }
81         }
82     }
83 }
84
85 @Override
86 public void onCancelled(DatabaseError databaseError) {
87
88 }
89 });
90
91 eventsSearch.setOnClickListener(new View.OnClickListener() {
92     @Override
93     public void onClick(View view) {
94         eventsSearch.setIconified(false);
95     }
96 });
97 }
98
99 //used for tests
100 public int getCountAdapter(){
101     return adapter.getCount();
102 }
103
104 @Override
105 public boolean onCreateOptionsMenu(Menu menu) {
106     getMenuInflater().inflate(R.menu.menu_main, menu);
107     return true;
108 }
109
110 @Override
111 public boolean onOptionsItemSelected(MenuItem item) {
112     int id = item.getItemId();
113     return (id == R.id.action_settings) || super.onOptionsItemSelected(item);
114 }
115 }

```

Code-Anhang 26: Events

EditEvent

```

1 public class EditEvent extends AppCompatActivity {
2
3     FirebaseDatabase mDatabase = FirebaseDatabase.getInstance();
4     DatabaseReference mRef = mDatabase.getReference("Events/");
5     DatabaseReference mRefBE = mDatabase.getReference("Benutzer-Events/");
6
7     boolean exec = false;

```

```
8 String nameID, eventID, invitedUsers;
9 ArrayList listEvents;
10
11 FirebaseAuth mAuth;
12
13 TextView editEventName, editEventCreator, editEventMembers, editEventDeadline;
14 EditText editEventPurposeText, editEventPlanText, editEventPlaceText,
15 editEventPriceText, editEventInfoText;
16 ImageButton editEventInviteMembersBtn, editEventDeleteBtn, editEventLeaveBtn,
    ↪ editEventSaveBtn;
17
18 @Override
19 protected void onCreate(Bundle savedInstanceState) {
20     super.onCreate(savedInstanceState);
21     requestWindowFeature(Window.FEATURE_NO_TITLE);
22     getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
    ↪ WindowManager.LayoutParams.FLAG_FULLSCREEN);
23     getWindow().setSoftInputMode(WindowManager.LayoutParams.
    ↪ SOFT_INPUT_STATE_HIDDEN);
24     setContentView(R.layout.activity_edit_event);
25
26     Bundle bundle = getIntent().getExtras();
27     if (bundle != null) nameID = bundle.getString("Name");
28     if (bundle != null) listEvents = bundle.getStringArrayList("listEvents");
29     if (bundle != null) invitedUsers = bundle.getString("members");
30
31     mAuth = FirebaseAuth.getInstance();
32     final FirebaseUser user = mAuth.getCurrentUser();
33
34     editEventCreator = findViewById(R.id.editEventCreator);
35     editEventName = findViewById(R.id.editEventName);
36     editEventMembers = findViewById(R.id.editEventMembers);
37     editEventDeadline = findViewById(R.id.editEventDeadline);
38     editEventPurposeText = findViewById(R.id.editEventPurposeText);
39     editEventPlanText = findViewById(R.id.editEventPlanText);
40     editEventPlaceText = findViewById(R.id.editEventPlaceText);
41     editEventPriceText = findViewById(R.id.editEventPriceText);
42     editEventInfoText = findViewById(R.id.editEventInfoText);
43     editEventInviteMembersBtn = findViewById(R.id.editEventInviteMembersBtn);
44     editEventSaveBtn = findViewById(R.id.editEventSaveBtn);
45     editEventDeleteBtn = findViewById(R.id.editEventDeleteBtn);
46     editEventLeaveBtn = findViewById(R.id.editEventLeaveBtn);
47
48     editEventMembers.setMovementMethod(new ScrollingMovementMethod());
49
50     mRef.addValueEventListener(new ValueEventListener() {
51         @Override
52         public void onDataChange(DataSnapshot dataSnapshot) {
53             for (final DataSnapshot ds : dataSnapshot.getChildren()) {
54                 String eventName = ds.child("Name").getValue(String.class);
55                 final String eventID = ds.child("EventID").getValue(String.class)
    ↪ );
56
57                 assert eventID != null;
58                 if (eventName != null) {
59                     if (eventName.equals(nameID)) {
60                         if (user != null) {
61                             final String email = user.getEmail();
62                             assert email != null;
63
64                             if (email.equals(ds.child("Creator").getValue(String
```

```

65         ↪ .class))) {
        // Add/Remove users from edit mode is not
        ↪ possible. Button has no functionality yet
        ↪ and it is disabled!
66        // Button add/remove users shows on only if the
        ↪ admin is editing the event.
67        editEventInviteMembersBtn.setVisibility(View.
        ↪ VISIBLE);
68        editEventInviteMembersBtn.setEnabled(false);
69
70        editEventDeleteBtn.setVisibility(View.VISIBLE);
71
72        editEventDeadline.setOnFocusChangeListener(new
        ↪ View.OnFocusChangeListener() {
73            @Override
74            public void onFocusChange(View view, boolean
        ↪ hasFocus) {
75                if (hasFocus) {
76                    DateDialog dialog = new DateDialog(
        ↪ view);
77                    FragmentTransaction ft =
        ↪ getFragmentManager().
        ↪ beginTransaction();
78                    dialog.show(ft, "Pick a date");
79                }
80            }
81        });
82    }else{
83        editEventLeaveBtn.setVisibility(View.VISIBLE);
84    }
85
86    editEventLeaveBtn.setOnClickListener(new View.
    ↪ OnClickListener() {
87        @Override
88        public void onClick(View view) {
89            AlertDialog.Builder alertDialogBuilder = new
        ↪ AlertDialog.Builder(EditEvent.this);
90            alertDialogBuilder.setTitle("Do you want to
        ↪ leave this event?\n");
91            alertDialogBuilder.setCancelable(false);
92
93            alertDialogBuilder.setPositiveButton("Yes",
        ↪ new DialogInterface.OnClickListener()
        ↪ {
94                @Override
95                public void onClick(DialogInterface arg0
        ↪ , int arg1) {
96
97                    mRef.addValueEventListener(new
        ↪ ValueEventListener() {
98                        @Override
99                        public void onDataChange(
        ↪ DataSnapshot dataSnapshot
        ↪ ) {
100                            ArrayList al = (ArrayList)
        ↪ ds.child("ListMembers
        ↪ ").getValue();
101                            assert al != null;
102                            for(int i = 0; i < al.size()
        ↪ ; i++){
103                                if(al.get(i).equals(

```



```
104         ↪ email)){
105         al.remove(i);
        mRef.child(eventID).
            ↪ child("
            ↪ ListMembers")
            ↪ .setValue(al)
            ↪ ;
106         mRefBE.child(eventID
            ↪ ).child(user.
            ↪ getUid()).
            ↪ removeValue()
            ↪ ;
107     }
108 }
109 }
110
111     @Override
112     public void onCancelled(
        ↪ DatabaseError
        ↪ databaseError) {
113
114     }
115 });
116
117     Intent i = new Intent(EditEvent.this
        ↪ , Events.class);
118     startActivity(i);
119     finish();
120 }
121 });
122
123     alertDialogBuilder.setNegativeButton("No",
        ↪ new DialogInterface.OnClickListener()
        ↪ {
124         @Override
125         public void onClick(DialogInterface
            ↪ dialog, int which) {
126         }
127     });
128
129     final AlertDialog alertDialog =
        ↪ alertDialogBuilder.create();
130     alertDialog.setOnShowListener( new
        ↪ DialogInterface.OnShowListener() {
131         @Override
132         public void onShow(DialogInterface arg0)
            ↪ {
133             alertDialog.getButton(AlertDialog.
                ↪ BUTTON_NEGATIVE).setTextColor
                ↪ (Color.RED);
134             alertDialog.getButton(AlertDialog.
                ↪ BUTTON_POSITIVE).setTextColor
                ↪ (Color.RED);
135         }
136     });
137
138     alertDialog.show();
139 }
140 });
141
142     editEventDeleteBtn.setOnClickListener(new View.
```

```
143         ↪ OnClickListener() {
144             @Override
145             public void onClick(View view) {
146
147                 AlertDialog.Builder alertDialogBuilder = new
148                     ↪ AlertDialog.Builder(EditEvent.this);
149                 alertDialogBuilder.setTitle("Do you want to
150                     ↪ delete this event?\n");
151                 alertDialogBuilder.setCancelable(false);
152
153                 alertDialogBuilder.setPositiveButton("Yes",
154                     ↪ new DialogInterface.OnClickListener()
155                     ↪ {
156                     @Override
157                     public void onClick(DialogInterface arg0
158                         ↪ , int arg1) {
159                         mRef.addValueEventListener(new
160                             ↪ ValueEventListener() {
161                             @Override
162                             public void onDataChange(
163                                 ↪ DataSnapshot dataSnapshot
164                                 ↪ ) {
165                                 mRef.child(eventID).
166                                     ↪ removeValue();
167
168                                 mRefBE.addValueEventListener
169                                     ↪ (new
170                                     ↪ ValueEventListener()
171                                     ↪ {
172                                     @Override
173                                     public void onDataChange
174                                         ↪ (DataSnapshot
175                                         ↪ dataSnapshot) {
176                                         mRefBE.child(eventID
177                                             ↪ ).removeValue
178                                             ↪ ();
179                                     }
180
181                                     @Override
182                                     public void onCancelled(
183                                         ↪ DatabaseError
184                                         ↪ databaseError) {
185                                     }
186                                 });
187                             }
188
189                             @Override
190                             public void onCancelled(
191                                 ↪ DatabaseError
192                                 ↪ databaseError) {
193                             }
194                         });
195                     Intent i = new Intent(EditEvent.this
196                         ↪ , Events.class);
197                     startActivity(i);
198                     finish();
199                 }
200             });
201         }
```

```
182         alertDialogBuilder.setNegativeButton("No",
183             ↪ new DialogInterface.OnClickListener()
184             ↪ {
185                 @Override
186                 public void onClick(DialogInterface
187                     ↪ dialog, int which) {
188                     }
189             });
190
191         final AlertDialog alertDialog =
192             ↪ alertDialogBuilder.create();
193         alertDialog.setOnShowListener( new
194             ↪ DialogInterface.OnShowListener() {
195                 @Override
196                 public void onShow(DialogInterface arg0)
197                     ↪ {
198                     alertDialog.getButton(AlertDialog.
199                         ↪ BUTTON_NEGATIVE).setTextColor
200                         ↪ (Color.RED);
201                     alertDialog.getButton(AlertDialog.
202                         ↪ BUTTON_POSITIVE).setTextColor
203                         ↪ (Color.RED);
204                 }
205             });
206
207         alertDialog.show();
208     }
209 });
210
211 String name = ds.child("Name").getValue(String.class
212     ↪ );
213 editEventName.setText(name);
214 String creator = ds.child("Creator").getValue(String
215     ↪ .class);
216 editEventCreator.setText(creator);
217 String members = ds.child("Members").getValue(String
218     ↪ .class);
219 editEventMembers.setText(members);
220 String deadline = ds.child("Deadline").getValue(
221     ↪ String.class);
222 editEventDeadline.setText(deadline);
223 String purpose = ds.child("Purpose").getValue(String
224     ↪ .class);
225 editEventPurposeText.setText(purpose);
226 String plan = ds.child("Plan").getValue(String.class
227     ↪ );
228 editEventPlanText.setText(plan);
229 String place = ds.child("Place").getValue(String.
230     ↪ class);
231 editEventPlaceText.setText(place);
232 String price = ds.child("Price").getValue(String.
233     ↪ class);
234 editEventPriceText.setText(price);
235 String moreInfo = ds.child("Information").getValue(
236     ↪ String.class);
237 editEventInfoText.setText(moreInfo);
238 EditEvent.this.eventID = ds.child("EventID").
239     ↪ getValue(String.class);
240 break;
241 }
242 }
```

```

223     }
224     }
225 }
226
227 @Override
228 public void onCancelled(DatabaseError databaseError) {
229
230 }
231 });
232
233 editEventInviteMembersBtn.setOnClickListener(new View.OnClickListener() {
234     @Override
235     public void onClick(View view) {
236         Intent i = new Intent(EditEvent.this, InviteMembers.class);
237         startActivity(i);
238     }
239 });
240
241 editEventSaveBtn.setOnClickListener(new View.OnClickListener() {
242     @Override
243     public void onClick(View view) {
244         saveEditEvent();
245         Toast.makeText(EditEvent.this, "Event was edited", Toast.
246             ↪ LENGTH_SHORT).show();
247         Intent i = new Intent(EditEvent.this, Mainpage.class);
248         startActivity(i);
249     }
250 });
251 }
252
253 private void saveEditEvent() {
254     mRef.addValueEventListener(new ValueEventListener() {
255         @Override
256         public void onDataChange(DataSnapshot dataSnapshot) {
257             for (DataSnapshot ds : dataSnapshot.getChildren()) {
258                 String nameKey = ds.child("Name").getValue(String.class);
259                 if (nameKey != null) {
260                     if ((nameKey.equals(nameID)) && !exec) {
261                         String eventDeadline = editEventDeadline.getText().
262                             ↪ toString();
263                         mRef.child(eventID).child("Deadline").setValue(
264                             ↪ eventDeadline);
265                         String eventPurpose = editEventPurposeText.getText().
266                             ↪ toString();
267                         mRef.child(eventID).child("Purpose").setValue(
268                             ↪ eventPurpose);
269                         String eventPlan = editEventPlanText.getText().toString
270                             ↪ ();
271                         mRef.child(eventID).child("Plan").setValue(eventPlan);
272                         String eventPlace = editEventPlaceText.getText().
273                             ↪ toString();
274                         mRef.child(eventID).child("Place").setValue(eventPlace);
275                         String eventPrice = editEventPriceText.getText().
276                             ↪ toString();
277                         mRef.child(eventID).child("Price").setValue(eventPrice);
278                         String eventInfo = editEventInfoText.getText().toString
279                             ↪ ();
280                         mRef.child(eventID).child("Information").setValue(
281                             ↪ eventInfo);
282                         exec = true;
283                     }
284                 }
285             }
286         }
287     });
288 }

```

```
274         }
275     }
276 }
277
278     @Override
279     public void onCancelled(DatabaseError databaseError) {
280
281     }
282 });
283 }
284
285     @Override
286     public boolean onCreateOptionsMenu(Menu menu) {
287         getMenuInflater().inflate(R.menu.menu_main, menu);
288         return true;
289     }
290
291     @Override
292     public boolean onOptionsItemSelected(MenuItem item) {
293         int id = item.getItemId();
294         return (id == R.id.action_settings) || super.onOptionsItemSelected(item);
295     }
296
297     public void onBackPressed(){
298         // user can only save the changes or use the menu to go back.
299     }
300 }
```

Code-Anhang 27: Event bearbeiten

Search

```
1 public class Search extends AppCompatActivity {
2
3     FirebaseDatabase mDatabase = FirebaseDatabase.getInstance();
4     DatabaseReference mRefB = mDatabase.getReference("Benutzer/");
5
6     SearchView searchUser;
7     ListView searchUserListView;
8
9     String userEmail;
10    ArrayAdapter<String> adapter;
11    ArrayList<String> allUsersList = new ArrayList<>();
12
13    @Override
14    protected void onCreate(Bundle savedInstanceState) {
15        super.onCreate(savedInstanceState);
16        requestWindowFeature(Window.FEATURE_NO_TITLE);
17        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
18            ↳ WindowManager.LayoutParams.FLAG_FULLSCREEN);
19        getWindow().setSoftInputMode(WindowManager.LayoutParams.
20            ↳ SOFT_INPUT_STATE_HIDDEN);
21        setContentView(R.layout.activity_search);
22
23        searchUser = findViewById(R.id.searchUser);
24        searchUserListView = findViewById(R.id.searchUserListView);
25
26        adapter = new ArrayAdapter<>(this, android.R.layout.
27            ↳ simple_list_item_activated_1, allUsersList);
```

```
25 searchUserListView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
26 searchUserListView.setItemChecked(1, true);
27 searchUserListView.setAdapter(adapter);
28
29 searchUserListView.setOnItemClickListener(new AdapterView.
    ↳ onItemClickListener() {
30     @Override
31     public void onItemClick(AdapterView<?> adapterView, View view, int i,
        ↳ long l) {
32         for (int nr = 0; nr < allUsersList.size(); nr++) {
33             userEmail = searchUserListView.getItemAtPosition(i).toString();
34             if (nr == i) {
35                 Intent intent = new Intent(Search.this, UserInfo.class);
36                 intent.putExtra("UserEmail", userEmail);
37                 startActivityForResult(intent, 9624);
38                 break;
39             }
40         }
41     }
42 });
43
44 mRefB.addValueEventListener(new ValueEventListener() {
45     @Override
46     public void onDataChange(DataSnapshot dataSnapshot) {
47         for (DataSnapshot ds : dataSnapshot.getChildren()) {
48             String email = ds.child("Email").getValue(String.class);
49             allUsersList.add(email);
50             adapter.notifyDataSetChanged();
51         }
52     }
53
54     @Override
55     public void onCancelled(DatabaseError databaseError) {
56
57     }
58 });
59
60 searchUser.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
61     @Override
62     public boolean onQueryTextSubmit(String text) {
63         return false;
64     }
65
66     @Override
67     public boolean onQueryTextChange(String newText) {
68         adapter.getFilter().filter(newText);
69         return false;
70     }
71 });
72
73 searchUser.setOnClickListener(new View.OnClickListener() {
74     @Override
75     public void onClick(View view) {
76         searchUser.setIconified(false);
77     }
78 });
79 }
80
81 //used for android tests
82 public int getSizeAdapter(){
83     return adapter.getCount();
```

```
84     }
85
86     @Override
87     public boolean onCreateOptionsMenu(Menu menu) {
88         getMenuInflater().inflate(R.menu.menu_main, menu);
89         return true;
90     }
91
92     @Override
93     public boolean onOptionsItemSelected(MenuItem item) {
94         int id = item.getItemId();
95         return (id == R.id.action_settings) || super.onOptionsItemSelected(item);
96     }
97 }
```

Code-Anhang 28: Suche

UserInfo

```
1 public class UserInfo extends AppCompatActivity {
2
3     FirebaseDatabase mDatabase = FirebaseDatabase.getInstance();
4     DatabaseReference mRefB = mDatabase.getReference("Benutzer/");
5
6     String userEmail;
7     TextView userInfoEmail, userInfoName, userInfoTelephoneText,
8         ↪ userInfoLeitzeichenText,
9         userInfoDepartmentText, userInfoRoomText, userInfoAdditionalText;
10    ImageButton userInfoCloseBtn;
11
12    @Override
13    protected void onCreate(Bundle savedInstanceState) {
14        super.onCreate(savedInstanceState);
15        requestWindowFeature(Window.FEATURE_NO_TITLE);
16        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
17            ↪ WindowManager.LayoutParams.FLAG_FULLSCREEN);
18        getWindow().setSoftInputMode(WindowManager.LayoutParams.
19            ↪ SOFT_INPUT_STATE_HIDDEN);
20        setContentView(R.layout.activity_user_info);
21
22        Bundle bundle = getIntent().getExtras();
23        if (bundle != null) userEmail = bundle.getString("UserEmail");
24
25        //Photo is an example. No update to the photo is possible yet!
26        userInfoEmail = findViewById(R.id.userInfoEmail);
27        userInfoName = findViewById(R.id.userInfoName);
28        userInfoTelephoneText = findViewById(R.id.userInfoTelephoneText);
29        userInfoLeitzeichenText = findViewById(R.id.userInfoLeitzeichenText);
30        userInfoDepartmentText = findViewById(R.id.userInfoDepartmentText);
31        userInfoRoomText = findViewById(R.id.userInfoRoomText);
32        userInfoAdditionalText = findViewById(R.id.userInfoAdditionalText);
33        userInfoCloseBtn = findViewById(R.id.userInfoCloseBtn);
34        loadUserInfo();
35        userInfoCloseBtn.setOnClickListener(new View.OnClickListener() {
36            @Override
37            public void onClick(View view) {
38                Intent i = new Intent(UserInfo.this, Search.class);
39                startActivity(i);
40            }
41        });
42    }
43 }
```

```

38     });
39 }
40
41 private void loadUserInfo() {
42     mRefB.addValueEventListener(new ValueEventListener() {
43         @Override
44         public void onDataChange(DataSnapshot dataSnapshot) {
45             for (final DataSnapshot ds : dataSnapshot.getChildren()) {
46                 String emailUser = ds.child("Email").getValue(String.class);
47                 assert emailUser != null;
48                 if(emailUser.equals(userEmail)){
49
50                     String email = ds.child("Email").getValue(String.class);
51                     userInfoEmail.setText(email);
52                     String name = ds.child("Name").getValue(String.class);
53                     userInfoName.setText(name);
54                     String leitzeichen = ds.child("Leitzeichen").getValue(String
55                         ↪ .class);
56                     userInfoLeitzeichenText.setText(leitzeichen);
57                     String room = ds.child("Raum").getValue(String.class);
58                     userInfoRoomText.setText(room);
59                     String tel = ds.child("Telefon").getValue(String.class);
60                     userInfoTelephoneText.setText(tel);
61                     String department = ds.child("Abteilung").getValue(String.
62                         ↪ class);
63                     userInfoDepartmentText.setText(department);
64                     String info = ds.child("Additional Info").getValue(String.
65                         ↪ class);
66                     userInfoAdditionalText.setText(info);
67                     break;
68                 }
69             }
70         }
71     });
72 }
73
74
75 @Override
76 public boolean onCreateOptionsMenu(Menu menu) {
77     getMenuInflater().inflate(R.menu.menu_main, menu);
78     return true;
79 }
80
81 @Override
82 public boolean onOptionsItemSelected(MenuItem item) {
83     int id = item.getItemId();
84     return (id == R.id.action_settings) || super.onOptionsItemSelected(item);
85 }
86
87 public void onBackPressed(){
88     // user can only go back with the additional back button or use the menu to
89     ↪ go back.
90 }

```

Code-Anhang 29: Suche Benutzerinformationen

D Abbildungen

Abbildung 3: Screenshots: Splashscreen, Startseite, Registrierung

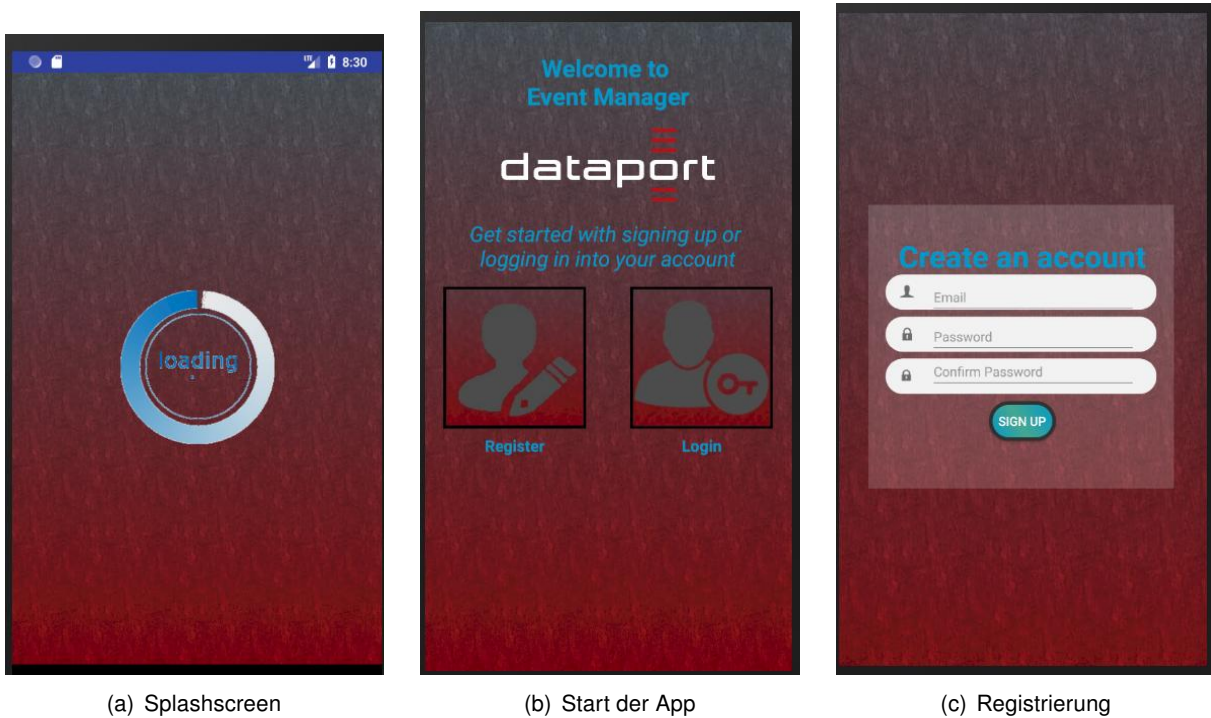


Abbildung 4: Screenshots: Anmeldung, Hochladen vom Bild, Profileseite mit FAB

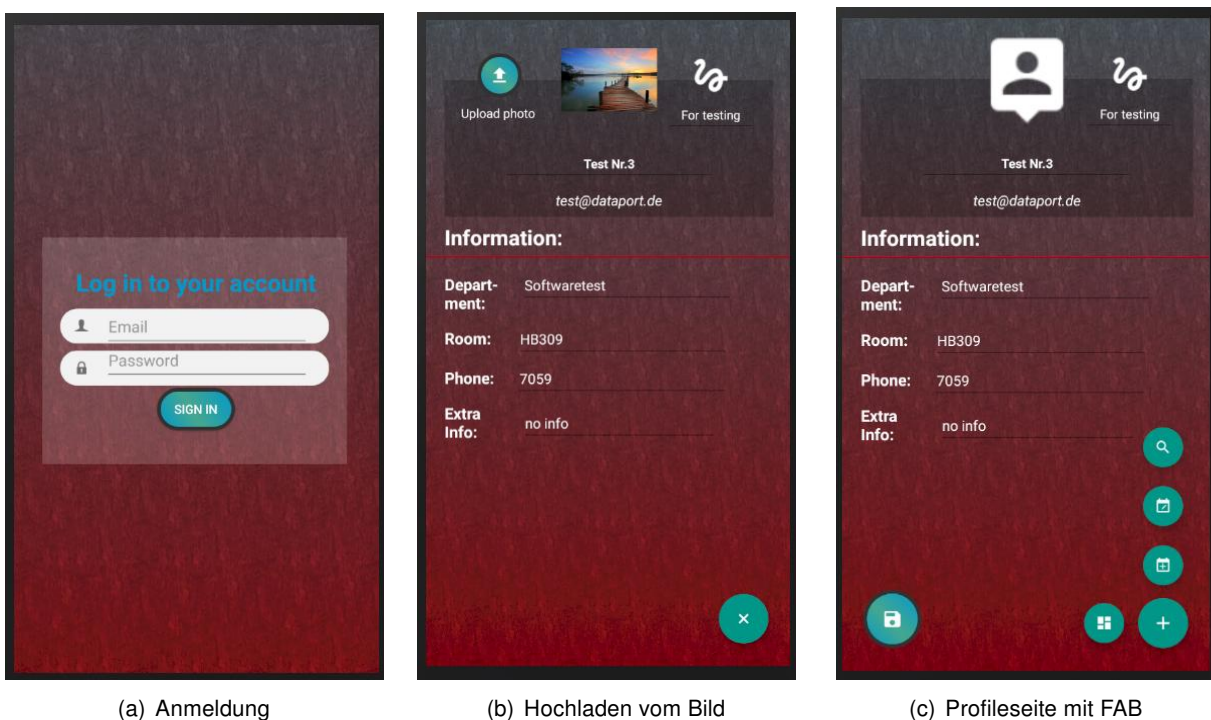


Abbildung 5: Screenshots: Event erstellen, Frist ansetzen, Teilnehmer einladen

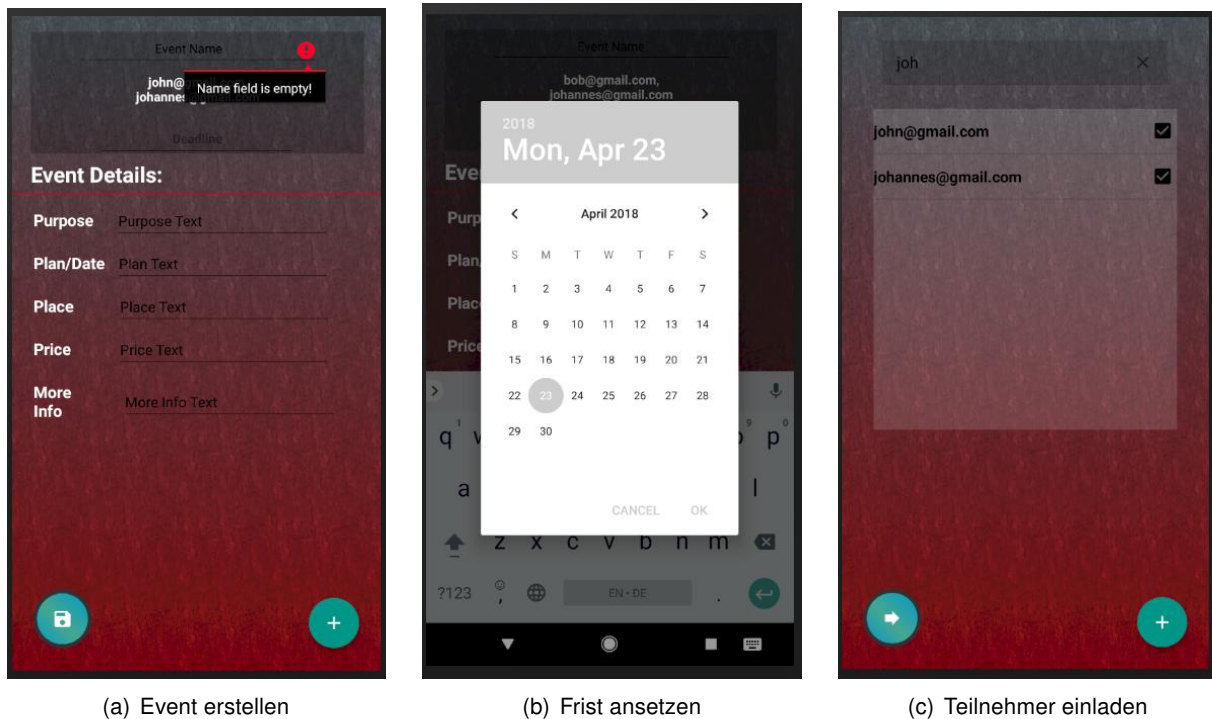


Abbildung 6: Screenshots: Event als Admin/Teilnehmer bearbeiten, Event verlassen

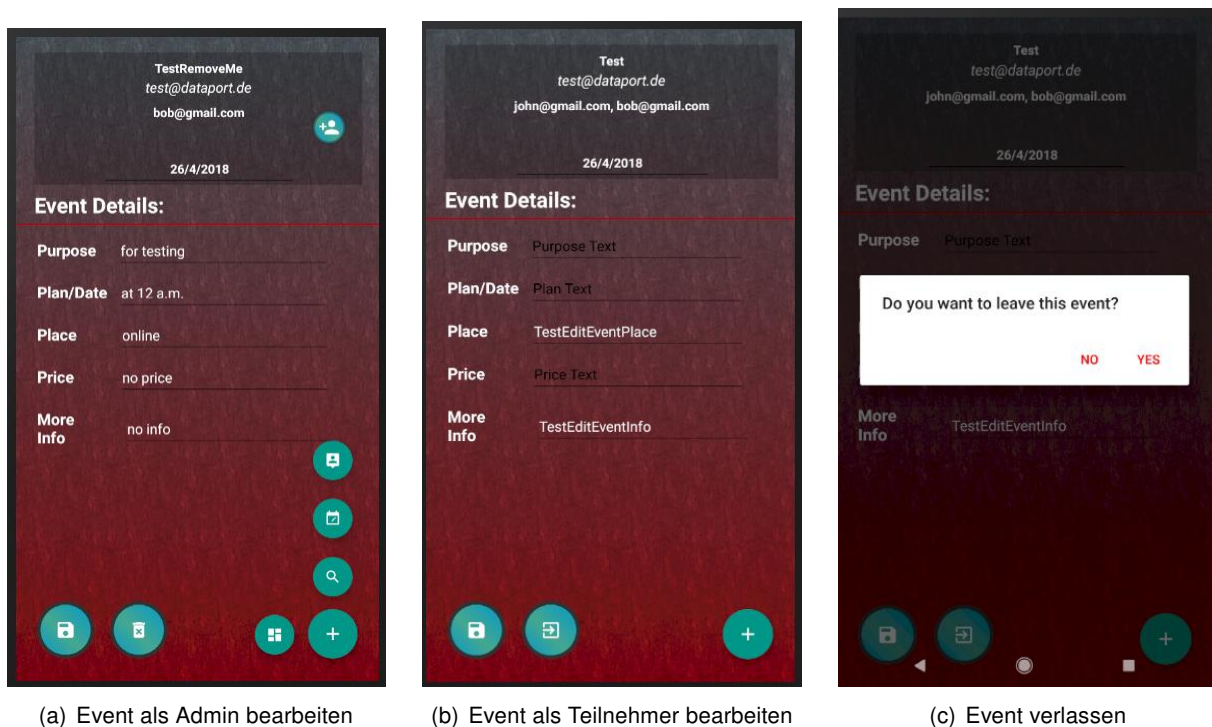


Abbildung 7: Screenshots: Event löschen, Mitarbeiter suchen, Visitenkarte

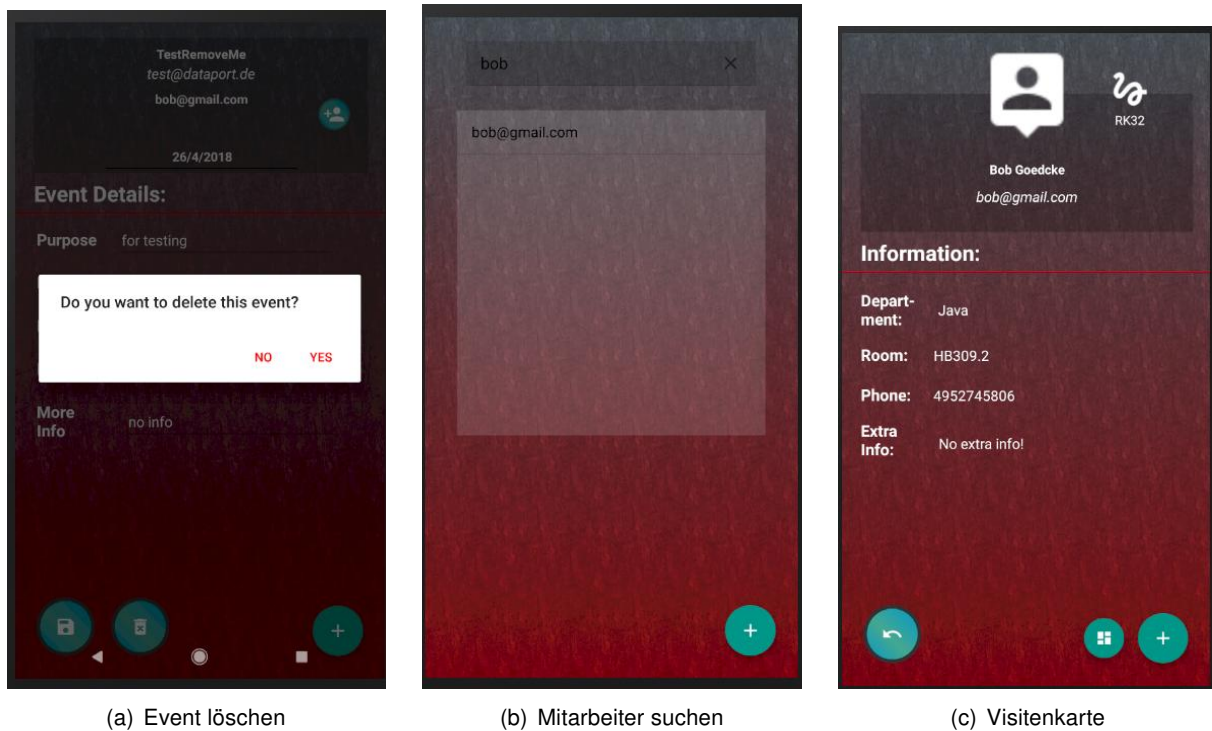



Abbildung 8: Screenshots: Hintergrundbild, Mockup Beispiel, Mockup Realisierung




Abbildung 9: Screenshots: Firebase TestLab - Robo-Test, alle erzeugte Tests


Robo-Test, vor 7 Tagen [ⓘ]

Fehlgeschlagen	Bestanden	Übersprungen	Nicht eindeutig
0	1	0	0

[SCREENSHOT-CLUSTER ANSEHEN →](#)

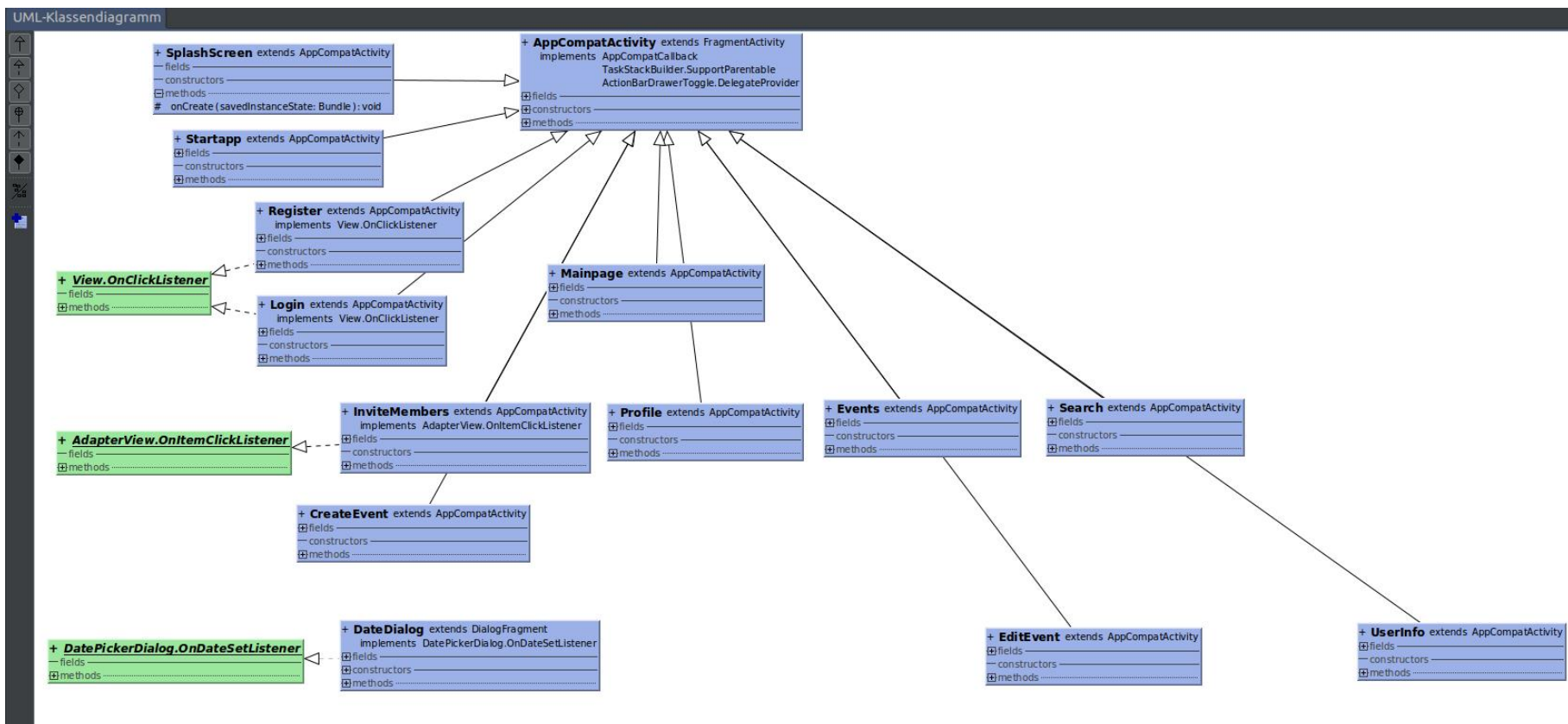
Testausführung	Dauer	Sprache	Ausrichtung	Probleme
 Pixel, API-Ebene 26	56 s	Englisch (Vereinigte Staaten)	Hochformat	—

(a) Firebase TestLab - Robo-Test

The screenshot shows the 'Run' tab in Android Studio, specifically the 'Test Results' section. The interface is dark-themed. At the top, there's a toolbar with icons for running, debugging, and other actions. Below the toolbar, the 'Test Results' list is expanded, showing a hierarchy of test classes and methods. Each item is preceded by a green circle with 'OK' inside, indicating successful test execution. The total execution time for all tests is 1m 35s 532ms. The test 'com.example.tonyk.promac.ProfileTest' is highlighted in blue. The test 'com.example.tonyk.promac.RegisterTest' is also expanded, showing sub-tests like 'emailExistsRegistrationFailed' and 'registration'. The test 'com.example.tonyk.promac.SearchTest' is expanded, showing sub-tests like 'testFAB', 'searchUser', and 'searchNotFoundUser'. The test 'com.example.tonyk.promac.StartappTest' is expanded, showing sub-tests like 'testGoToRegisterBtn' and 'testGoToLoginBtn'. The test 'com.example.tonyk.promac.UserInfoTest' is expanded, showing sub-tests like 'testGoToRegisterBtn' and 'testGoToLoginBtn'.

Test Class	Test Method	Duration
com.example.tonyk.promac.CreateEventTest		2s 461ms
com.example.tonyk.promac.EditEventTest		1s 433ms
com.example.tonyk.promac.EventsTest		22s 251ms
com.example.tonyk.promac.EventsTest	testFAB	1s 433ms
com.example.tonyk.promac.EventsTest	editEvent	7s 405ms
com.example.tonyk.promac.EventsTest	testFAB	1s 156ms
com.example.tonyk.promac.EventsTest	adminToTheEvent	3s 74ms
com.example.tonyk.promac.EventsTest	searchEvent	2s 637ms
com.example.tonyk.promac.EventsTest	removeEvent	5s 410ms
com.example.tonyk.promac.EventsTest	notAdminToTheEvent	2s 569ms
com.example.tonyk.promac.ExampleInstrumentedTest		0ms
com.example.tonyk.promac.ExampleInstrumentedTest	useAppContext	0ms
com.example.tonyk.promac.InviteMembersTest		5s 768ms
com.example.tonyk.promac.InviteMembersTest	nobodyInvited	1s 461ms
com.example.tonyk.promac.InviteMembersTest	testFAB	1s 259ms
com.example.tonyk.promac.InviteMembersTest	inviteSomeone	3s 48ms
com.example.tonyk.promac.LoginTest		28s 628ms
com.example.tonyk.promac.LoginTest	login	682ms
com.example.tonyk.promac.LoginTest	passwordIsEmpty	5s 403ms
com.example.tonyk.promac.LoginTest	emailsRequired	4s 138ms
com.example.tonyk.promac.LoginTest	loginFailed	6s 562ms
com.example.tonyk.promac.LoginTest	passwordIsTooSmall	6s 392ms
com.example.tonyk.promac.LoginTest	emailsInvalid	5s 451ms
com.example.tonyk.promac.MainpageTest		2s 473ms
com.example.tonyk.promac.MainpageTest	logOut	732ms
com.example.tonyk.promac.MainpageTest	goToProfile	378ms
com.example.tonyk.promac.MainpageTest	goToCreateEvent	427ms
com.example.tonyk.promac.MainpageTest	goToEvents	630ms
com.example.tonyk.promac.MainpageTest	goToSearch	306ms
com.example.tonyk.promac.ProfileTest		6s 503ms
com.example.tonyk.promac.RegisterTest		16s 162ms
com.example.tonyk.promac.RegisterTest	emailExistsRegistrationFailed	7s 891ms
com.example.tonyk.promac.RegisterTest	registration	8s 271ms
com.example.tonyk.promac.SearchTest		1s 338ms
com.example.tonyk.promac.SearchTest	testFAB	453ms
com.example.tonyk.promac.SearchTest	searchUser	403ms
com.example.tonyk.promac.SearchTest	searchNotFoundUser	482ms
com.example.tonyk.promac.StartappTest		5s 955ms
com.example.tonyk.promac.StartappTest	testGoToRegisterBtn	2s 462ms
com.example.tonyk.promac.StartappTest	testGoToLoginBtn	3s 493ms
com.example.tonyk.promac.UserInfoTest		2s 560ms

Abbildung 10: Alle UML-Klassen



Toni Kozarev

