Ready2Meet –Android

Mobile Application Development with massive user support

Berkay Köksal

Spring 2018

Table of Contents

1. Application Description

Ready2Meet offers a free platform for people to socialise with others by participating events together! Important features are designed for you to create your own event and interact with people around you to participate together. The user creates an event (including a start and end time, location, type of event and eventually a name). In the planned event mode, the user can invite specific people (friends) while for the spontaneous event, it is not possible to invite specific people. Instead, everyone will be able to join the event which should only be displayed to a user if he is close to the event's location. Like this, the organizer of the event is able to spontaneously meet with people based on their location.

Once a user joined an event, she/he is able to share pictures of the event in the app and eventually influence the event's outcome e.g. by voting for music on a party. A chatroom is available to communicate with the people before or after the event and thus allow networking and spreading information to all participants. For all kinds of events, the organizing person can add a list of required material (e.g. drinks or food) so that the guests can register to bring some of the material and thus easing the organization.

Finally, every user invited to an event is provided additional information like the weather forecast for outdoor events and whether he is available on the date when the event is scheduled.

2. Business Model

Different business models are possible to push the app on the market. In this section, we discuss potential models and pricing strategies and nally chose the most promising model. Furthermore, we discuss a marketing strategy for our application.

2.1 Monetization

A completely free (e.g. open source) app is the first possibility. While this potentially targets the highest number of users as the app is free to use and accessible also in alternative stores like F-Droid, the application would not lead to any profit.

A second option is to use advertisements in the app in order to gain from the application. While this leads to a higher income, it contradicts the open-source strategy and thus would probably lead to a smaller target group. However, as the vast majority of Android users install apps from the Google Play Store, the decrease of users is acceptable. Together with the app using advertisement, it is possible to offer an for a small price and without ads. An app which has only a priced version does not appear to be promising to us as most users will not be willing to pay for the service.

In App purchases are used to boost the normal user constraints. In especial, this makes sense for business customers that could use the application to advertise their own business events. A possible scenario is creating events in the application which reflects the company's product portfolio (e.g guided tours, sports, parties in a bar, etc). Like this, the company can acquire new customers which are nearby e.g. during their holidays and thus our app presents a new marketing strategy. In the scenario of commercial users, it would be possible to earn a reward for every user which joins the respective event by our platform. The reward could be negotiated as a percentage of their ticket sales for the event or as a fixed amount. However, both cases would require the payment system to be outside of the application since GooglePay billing service only approves in app feature and product sales that are related to the application itself.

In our case, the most promising strategy was to offer a free app with advertisement as well as a paid version without ads (or im app purchase to remove ads). In-App sales target commercial customers and normal users who would like to make their influence more powerful than normal users. E.g. an event can be permanent, or it's radius can overreach the normal constraints of non paying users. Premium users are able to create more than 5 events that are in the future and make the best of the platform by reaching more audience.

2.2 Marketing Strategy

As our app aims at making events available to the highest possible number of potential participants, organizing events is the main possibility to advertise the app to attract more users in the beginning.
First, it is possible to offer free events to users which are organized by us. Second, we could offer discounts to our users for paid events which are organized by a third party. This is beneficial for both, the organizing party as they get additional participants and us, as we can make our app attractive for many users and thus gain a high number of users.
Both techniques aim at attracting as many users as possible. Even if we have some expenses for the events and eventually for offering the discounts, this can decline once we have sufficient customers.

3. Features

The features of our event organization app include

- Fully integrated user authentication
    - Signup using email or Facebook
    - Reset password with mail
    - Change mail address
    - Delete account
    - Login/Logout

- Create an event (e.g. party, sports, hiking, lunch, …)
    - Manage people joining your event
        - Kick someone from the event
    - Cancel your event

- Chatroom for events and direct messaging between users
    - Use chat room to talk with participants
    - Or use direct messaging from user profile pages
    - UTF8 + Emoji support on chat room

- Share pictures of the event
    - Upload your pictures for the event
    - Download other pictures from the event

- Get information about nearby events
    - Date of events around you
    - Location of events
    - Weather forecast for outdoor events
    - Check if your calendar is free for this event
    - Check who is joining this event

- Fully integrated user profile management
    - Follow other user profiles
    - Message other users
    - See their profile for history of their events
    - Categories they are interested
    - Block users to not see them (vice versa)
    - Unblock users again from your account options

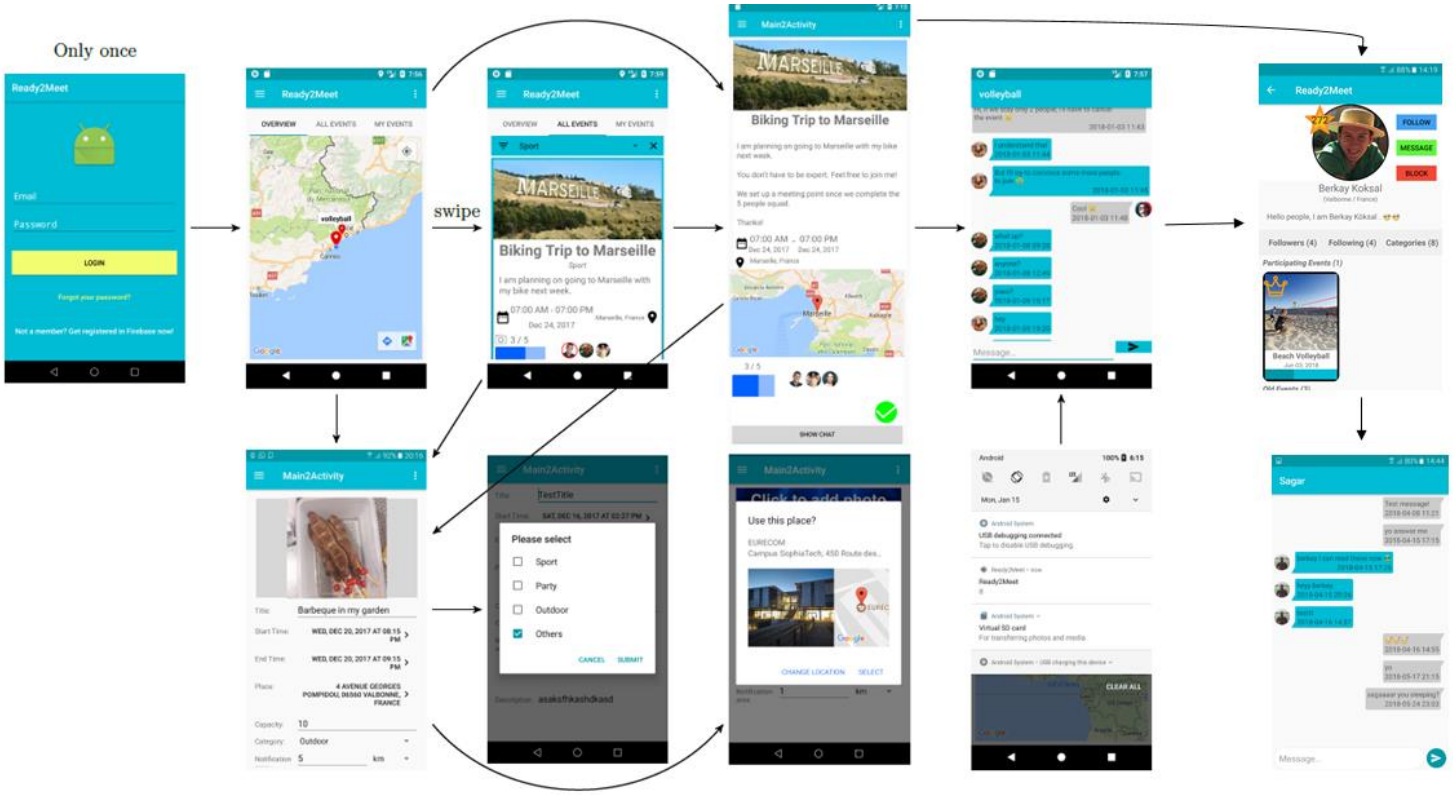
- Explore people around you
    - See who is around you, to decide who you should invite to your event

- Premium Subscription
    - 1€ a month in app purchase for premium account
    - Premium users can create more than 5 events
    - Premium users get a star on their profiles showing days left on premium
    - Premium users can remove ads (TODO)

- Share the event in other networks e.g. on facebook - Further development

4. UI Design and Click Stream

The app is designed such that every page can be reached with only 2 clicks after starting the app. The first screen of the app is the login page. Once a user has registered and logged in for the first time, the user credentials are stored so that this step will be skilled in further uses of the app.
Once the user is logged in, the start screen of the app is designed to provide an overview of interesting events for the user. We display all events the user might be interested in according to his/her interested categories declared through the profile. Also, the user's current location is used to show only events which are in his/her surrounding. To indicate the popularity of an event, we use the number of participants that will make the marker bigger as more people are joining. By clicking on the marker, the title of the event is shown. After clicking on title, the event details are displayed. This page allows the user a quick orientation and aims at targeted presentation of events to the user.
By swiping to the right from the start screen, the events are listed with increasing event time, meaning events that will start shortly will be printed on top of others. It is possible for the user to filter events according to predefined event categories. After filtering, the search is stored in shared preferences for later uses of the app, the categories interested are also stored in real time database for this particular user's data field. This allows us to reduce the user's clicks as we can assume that his preferences for event categories do not change that frequently. By clicking on one of the events, its details are shown.
The even details show a complete overview of the event. On the top, every participant of an event can add photos to an image gallery dedicated for this event. Participants can also download the images they like from the same place. Afterwards, event details like a description, time and location as well as conflicts with the own calendar are given to all users of the app. Also, all participants are shown and a bar indicates how many salots have already been occupied for this event. If the user participates on an event, he will get access to the chat room of the event while non-participants will never get to see this room.
A unique chat for every event can be used by all participants of an event to communicate before or after an event and thus discuss organizational details or share impressions. If new messages arrive in one of the user's chats, a notification is sent also if the app is closed and serves as a quick link to the chat room when clicked.
For every screen of the app (except the chat), the user can reach a screen to add new events with only one click. Several dialogs are used to make the creation of the event integrative and provide us with sanitized data. User input is always checked before being committed to the database.

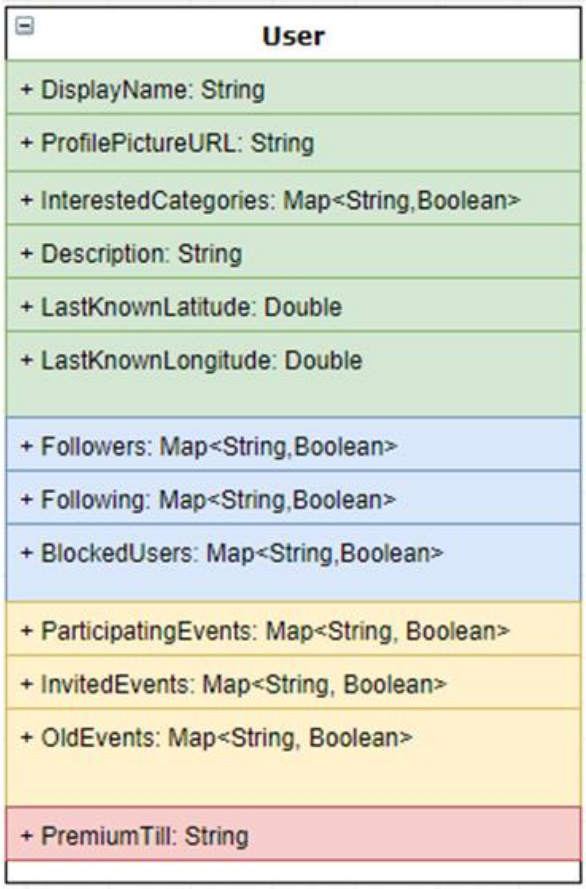

Only once
Ready2Meet
Email
Password
LOGIN
swipe
Biking Trip to Marseille
Please select
Sport
Party
Outdoor
Others
Use this place?
Barbeque in my garden
Berkay Koksal
Sagar

5. Implementation

In this section, we discuss technical details of our implementation.

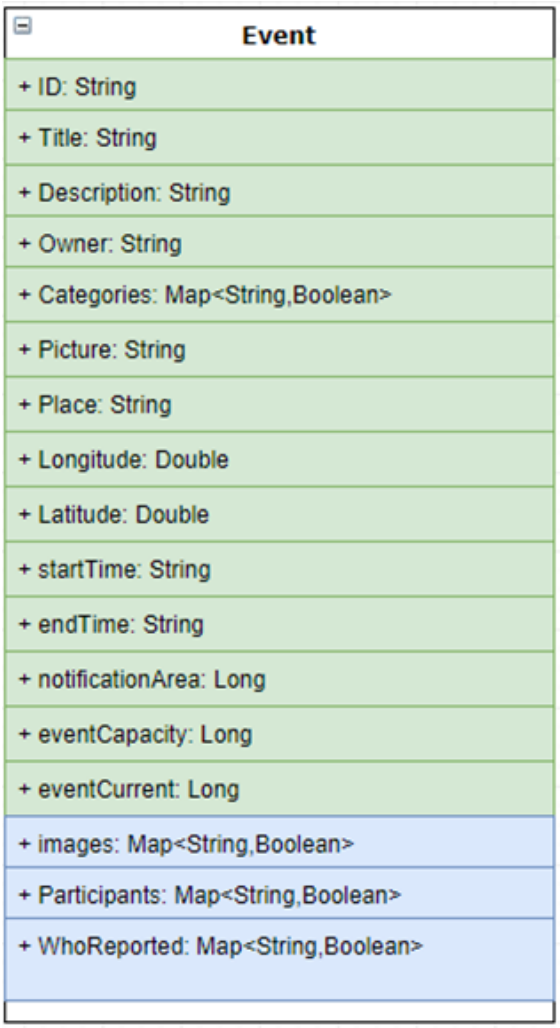
5.1 Database

To store our data (events, messages, users) and make them available to all users, we heavily make use of the features of Firebase database. Firebase provides a real-time database and is therefore ideally suitable to make changes on data available to multiple users in real time. Apart from that, it can easily be integrated into Android apps (as well as iOS and web applications). We need to store 3 different types of data structures namely user accounts, chat messages and events. For each of the structures, one java class denes the data elds which are stored.

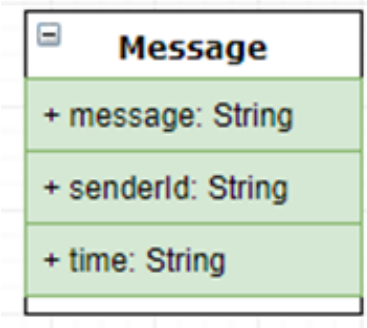


User data consumes the most space compared to other data entities. Every user should have a display name and a profile picture that distinguishes it from the others. Categories user is interested is kept, with a description of a profile page. Also every time user opens the app, last known location is written to the database to know where the user is. This information will be used to display events to the user. User entities will also keep track of followers and who is following this user, as well as the list of users blocked by that user.

User entities will also keep track of events this user is participating. The events that are passed are classified as OldEvents where future events are in ParticipatingEvents list. And finally, there is a PremiumTill field that keeps track of until which date this user has premium. When new premium is purchased, if there is already one it will add 30 days to it. If there is none, it will set it to today plus 30 days.



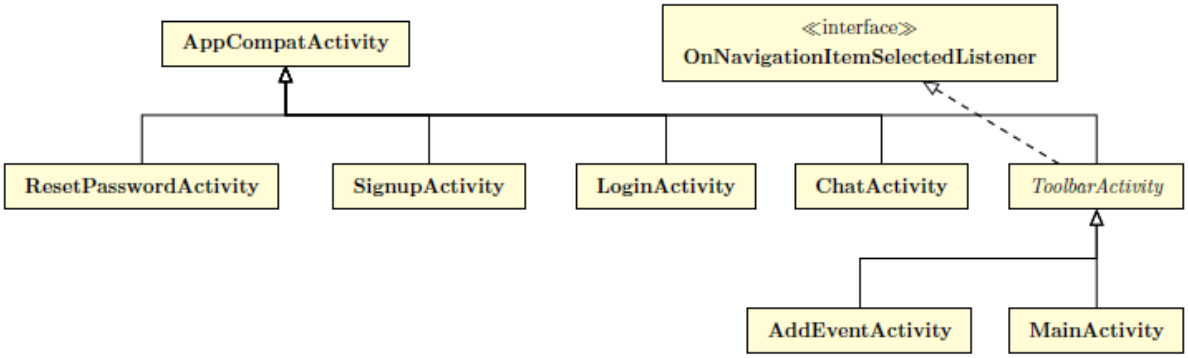
Events are kept similar to users. Ever event will have individual properties like title description place location start and end time notification area capacity and current number of participants. Important fields here are, Place string is the address of the location that the users see, where coordinates are the ones the application actually use to represent them. Owner keeps the ID of the user who creates the event, and is transferable to someone else. An event can have multiple categories. Event images are kept in a list of URLs to populate inside the listviews when required. Participant and WhoReported lists also keep userIDs relevant to these lists.



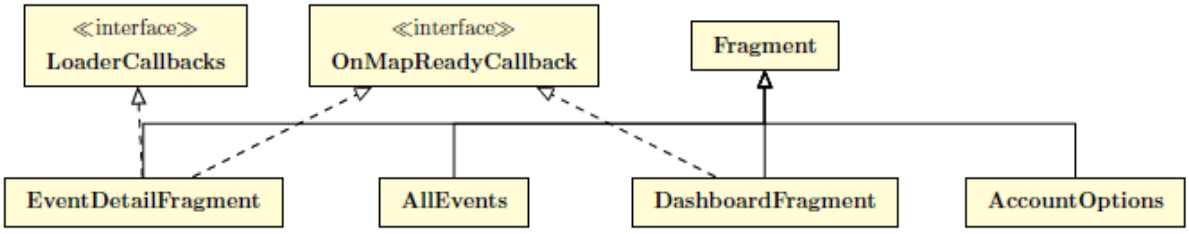
Message entity is the smallest in the database. However, in total it consumes the most storage in the database because of its excessive amount of usage. Every user can have one profile, maybe 5 events each. However, one user can send and receive thousands of messages. This is the reason we are not keeping isRead, isTransfered, DateofRead etc properties in this entity.

5.2 Application Components

To provide the functionality, we implemented several different components which we discuss in this section.. First, the three activities ResetPasswordActivity, SignupActivity, LoginActivity are responsible for managing user accounts and the corresponding login credentials. The class ChatActivity shows the chatroom for a selected event and requires the ID of the event in an extra in order to retrieve the chat messages. AddEventActivity is used to add one new event to the event database. The MainActivity shows the start screen of the app and displays all events on a map as well as in a list. To make our app more interactive, we have a couple of fragments. When the user starts the app, the DashboardFragment is shown which provides an overview over all events in a map. Each event has an own marker on the map. Different sizes of the markers are used to resemble the number of users of the corresponding events and thus allow the user to easily nd more popular events. Next, the fragment AllEvents shows the events in a list and allows ltering events with respect to their category (e.g. Sport, Outdoor, Party, . . .).



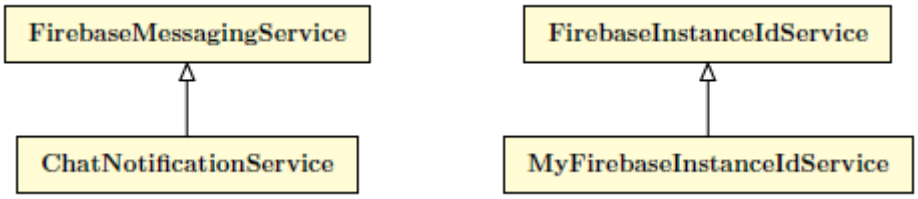
After clicking on an event in one of the two views, the EventDetailFragment is displayed which makes the full event details available to the users, including an image gallery, description, time and place of the event. Furthermore, possible collisions of the event with the calendar are checked to give the user feedback if it is likely for him to be available during the event. Also, the user can join and unjoin an event and access the chatroom if he registered for the event.



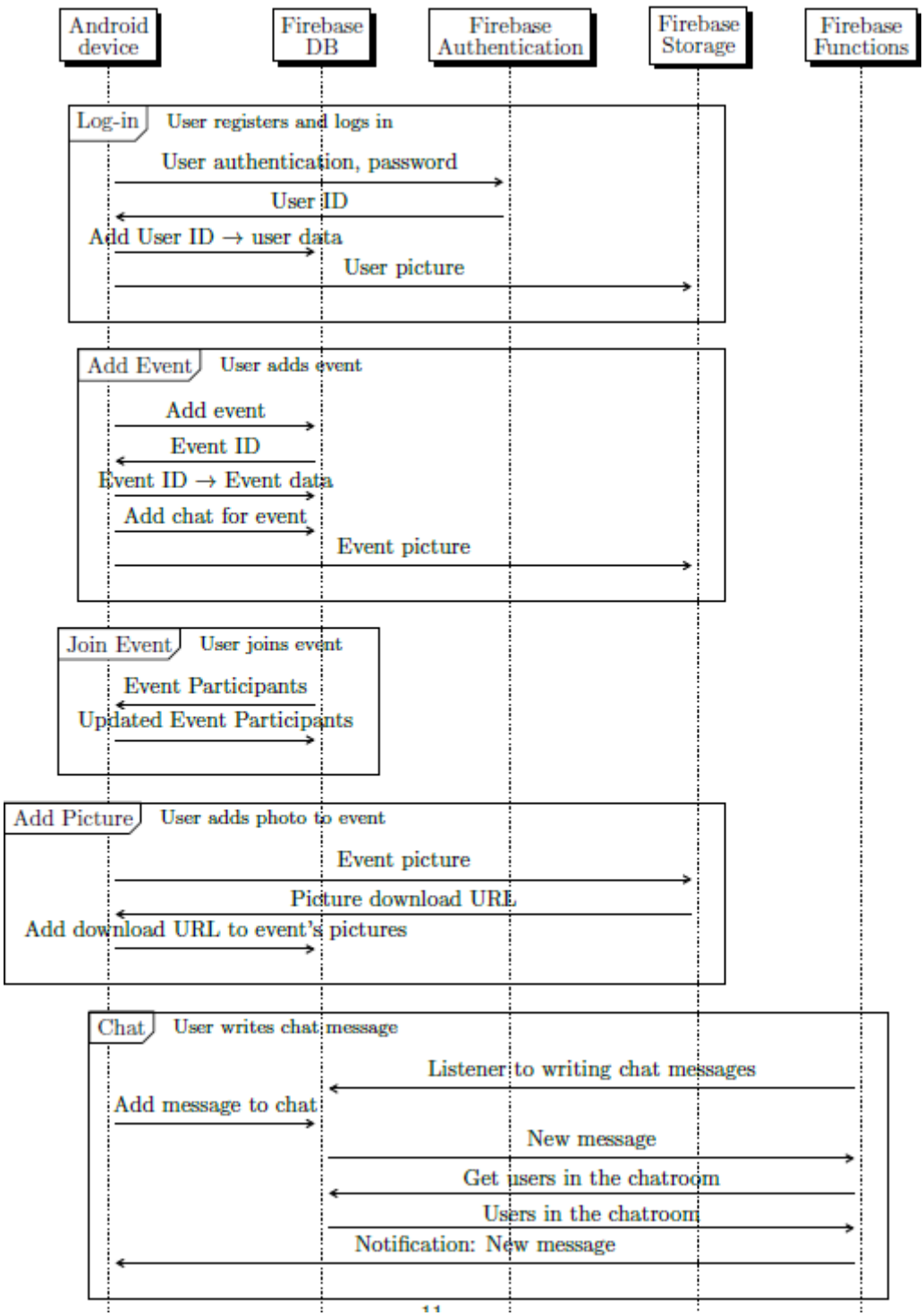
Two services are continuously running to give the user feedback on changes in the Firebase database. MyFirebaseInstanceIdService is used to keep the users' ID which is necessary for Firebase Cloud Messaging (FCM) up-to-date. The ChatNotificationService receives all notication messages from our FCM instance and displays them to the user. Like this, the user is always kept informed about new messages in the chats he joined.

5.3 Frontend/Backend Interaction

Our backend completely relies on Firebase and makes use of multiple of its different features.



First, the user authentication is based on Firebase's authentication features. This eases registering users, the log-in as well as advanced features like resetting passwords. We store additional data in the Firebase database to better address our users. Finally, every user has an own picture which is stored in Firebase storage. In order to add event, we rst make Firebase DB create a new ID for an event in the DB. For this event ID, the event data (i.e., title, description, time and place) are stored under the event ID. Finally, a folder for the event is created in Firebase storage and the event picture is uploaded to the folder. When additional pictures are added, they are stored in this folder and a value containing a link to download the image is added to the event instance in the DB. If a user joins an event, he fetches the participants of the event to join from the DB and sends an updated participant list to the DB. Finally, every user can write messages to a chat for each event he participates in. As we want to provide targeted push-notifications to users which are part of the chat, we implemented a Node.js component which is running as Firebase function instance (short function). The function is a listener to the DB and is notified whenever a new message is written to the chat. The new message is sent from the DB to the function which then retrieves the users of the corresponding chat and nally assembles a notification which is sent to all of these users. A service of the app receives the notification and displays the notification on the screen.


Android device
Firebase DB
Firebase Authentication
Firebase Storage
Firebase Functions
Log-in
User registers and logs in
User authentication, password
User ID
Add User ID → user data
User picture
Add Event
User adds event
Add event
Event ID
Event ID → Event data
Add chat for event
Event picture
Join Event
User joins event
Event Participants
Updated Event Participants
Add Picture
User adds photo to event
Event picture
Picture download URL
Add download URL to event's pictures
Chat
User writes chat message
Listener to writing chat messages
Add message to chat
New message
Get users in the chatroom
Users in the chatroom
Notification: New message