Internet-Praktikum: Telekooperation

Project: Sechzehn

Team Bravo: **Alexander Geiß** (alexanderhelmut.geiss@stud.tu-darmstadt.de),
Lukas Klein (lukas.klein@stud.tu-darmstadt.de),
Martin Lichtblau (martin.lichtblau@stud.tu-darmstadt.de),
Johannes Semsch (johannesmaximilianchristian.semsch@stud.tu-darmstadt.de),
Tim Walter (tim.walter.10@stud.tu-darmstadt.de)

TECHNISCHE UNIVERSITÄT DARMSTADT

Contents

1 Motivation

With the backing of its helpful and positive community, Sechzehn guides you through venues all around the world. Places like restaurants and museums are collectively rated. Each and every user of Sechzehn has the ability to share his personal experiences with the whole community. Our rating not only includes a textual rating system, it also allows the user to upload photos taken at this place. Furthermore we also offer a short like and dislike rating.

Sechzehn also has a social component. You see other members of our community on a map. This map combined with our real-time chat paves the way to quickly establish new and long-lasting friendships. Our real-time chats focuses on the most important features a messenger should have. This includes read receipts and the transmission of emojis. We do not distract you with features no one needs like photo filters and drawing capabilities in the chat. To satisfy more privacy-aware users who might not be happy with frequent position updates, we also provide an incognito mode. If the user enters it, he is not displayed on the map of other users. This feature makes it possible to use our app if the user is somewhere where he is not expected to be.

An intuitive and enjoyable user experience is important for us. That's why we designed our app following the material design guidelines.

2 Overview

In this section we present the architecture we used and thereby how our components interact with each other. We then briefly describe which technologies we chose, and why we chose them.

2.1 Architecture

2.1.1 Interaction of Components

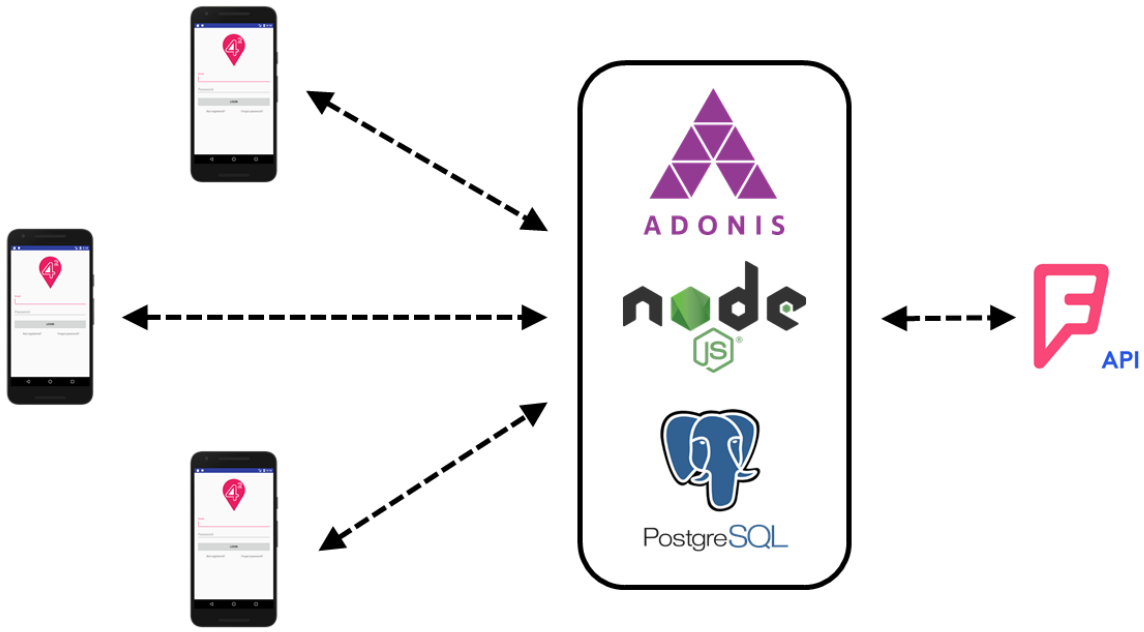

Figure 2.1: Architecture

The communication between the clients and our server is illustrated in Figure 2.1. For their communication with our server, the clients access the RESTful endpoints or use WebSocket connections. The latter one is necessary to provide our real-time chat. Since we do not have a huge community that backs our app, we need to retrieve the data from other sources. Our server therefore connects to the Foursquare API and fetches the venues, venue details and photos dynamically.

2.1.2 Server API Documentation

To connect app and server, the app developers need the knowledge of endpoints, query parameters and the data structures of requests and responses. To prevent that they need to look through the server's code, an extensive documentation is provided using Apiary [1]. An impression is given in Figure 2.2.

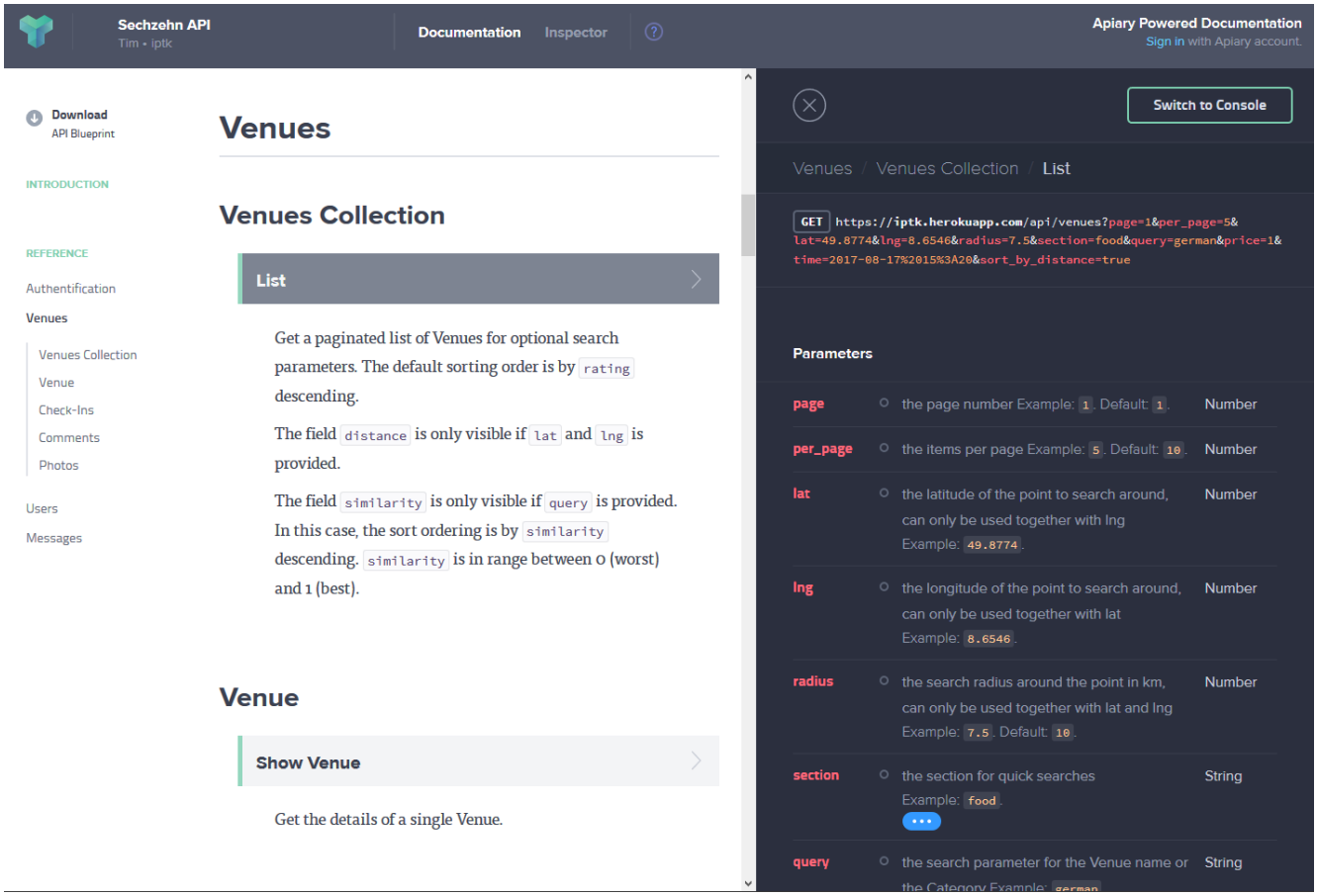

Figure 2.2: Screenshot of the Apiary Documentation

2.1.3 Database

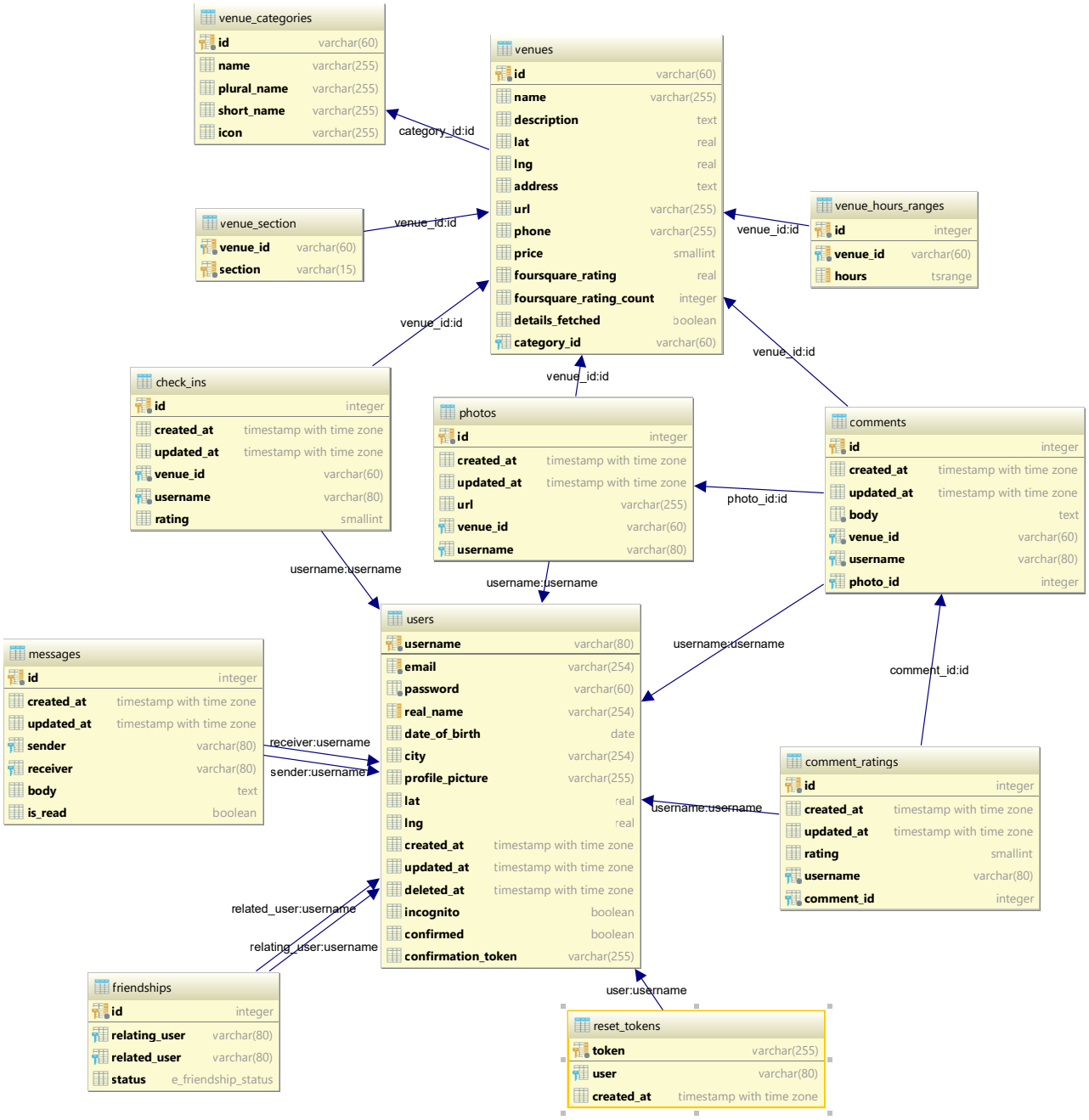

Figure 2.3: Database scheme

2.2 Server

To host our server we use the Heroku Cloud Application Platform [2]. Heroku provides us a PostgreSQL [6] database service and an Node.js execution environment. Furthermore it offers access to Cloudinary [3] image storage service that can be used as plugin within heroku. Heroku offers a free plan “hobby-dev” that we use.

2.2.1 Node.js

Node.js [4] is a cross-platform environment that allows building web servers with JavaScript. This is contrary to the usual applications that are written in JavaScript since it was firstly designed to be a client-side scripting language for browsers. Node.js in its current version also supports asynchronous execution and concepts like generator functions. This allows easy-readable code on a high abstraction level with high-order functions and a good handling of asynchronous operations.

2.2.2 AdonisJs

AdonisJs [5] is a MVC framework for Node.js web applications. It offers a further layer of abstraction and handles HTTP routing, requests and responses, content negotiation and an object-relational mapping to persist and fetch models from the database. In addition, it provides APIs for user input validation and sanitization, mail sending with templates, omitting fields from JSON output and authentication middleware. Due to this abstraction, it is not necessary to deal with unpretentious programming work that has been done multiple times before and is always the same for web application (e.g. routing, templating, authentication).

2.2.3 PostgreSQL

PostgreSQL [6] is a powerful open-source relational database management system. It provides the ability to add useful extensions, new data types and functions. We use extensions to determine if a venue or another user is near to the user's current position and to compute a similarity metric on two strings. The latter is the basis of the venue and user search. PostgreSQL offers data types for time ranges that allow to search for open venues. In this context, a custom function is used to convert opening hours into an unambiguous database representation that prevent overlaps. Fortunately, heroku provides a PostgreSQL plugin.

2.2.4 JSON Web Tokens

The authentication in our project is done with JSON Web Tokens (JWTs) [7]. Given a secure connection, JWTs offer a secure, stateless authentication methods that renders sessions and cookies unnecessary.

2.3 Client

2.3.1 Android Data Binding

Android Data Binding [8] enables us to write declarative layouts. Those can be easily adapted with new data in various forms by just passing a data object. Therefore it minimizes the amount of glue code needed. Furthermore it implements a kind of view holder pattern, so it is not necessary to use `findViewById` since all views are available through the binding object.

2.3.2 Retrofit

In our app HTTP requests are done with Retrofit[9]. Retrofit is a type-safe client for Android that helps the developer to retrieve and upload structured data like JSON via a REST based web service. Retrofit itself is based on the OkHttp Library. We chose to use Retrofit because it is a state-of-the-art technology that is also recommended by the Google Guidelines.

2.3.3 GSON

GSON[10] is a Java library that offers a simple way to serialize and deserialize JSON Objects to Java Objects and the other way around. One of the reason why we chose it, that it does not require additional class annotations like alternative libraries do. GSON is well documented and developed (and recommended) by Google.

2.3.4 ChatKit

ChatKit for Android [11] is a library that is designed to simplify the UI development of WebSocket based chats in Android. It offers predefined components like Dialog Lists, Message Lists, and Message Input. Furthermore it provides state-of-the-art messaging features like read receipts without too much additional effort.

3 App and Feature Explanation

3.1 System Requirements

To run the app with all features Android Marshmallow (API 23) is needed. In this section we briefly describe which permissions are requested by the app and why they are necessary.

1. Permission to take pictures and videos: This permission allows us to take pictures and add them to venues.
2. Permission to get access to pictures, media, and files on the device: Normally users use profile pictures for more than one app, what makes it necessary to load those pictures from the gallery. This permission is further needed to store the users login credentials on the phone.
3. Permission to have access to the location of the device: The whole concept of the app is based on the users location. The location is necessary to see users and venues in the surroundings. With an incognito switch however, the user can hide his account from the map.

3.2 Get Started

To install the app the user has to put the APK file from our git repo on his phone. The next step is installing the APK by tapping on it in the file explorer of your choice. If the users phone has not the permission to install apps from third parties, he has to allow the phone to install apps from third parties. This is described by the following steps (some phones will directly lead the user to step 3):

1. go to “Settings”,
2. choose “Security”,
3. scroll to “Unknown sources” and
4. allow the installation from unknown sources.

If the users phone is allowed to install apps from third parties, the phone asks the user to install the app. To do so he has to tap on install. The app is now installed successfully.

3.3 Features

We only list the bonus features because the mandatory features have to be fulfilled.

1. Account verification with an email:
   After registering the email has to be verified first. Without this verification, the user cannot use the app.
2. Open the links from the emails:
   After resetting your password or registering a new account the user receives an email containing a link. Our app is able to handle this link on click. This means that no verification code must be entered and no external browser is necessary to do this.
3. Resource-efficient location updating:
   To treat the users battery with care, we use the Google Location Services. If another app on the system requests the location via those location services, we use the new location if it is more accurate. If there hasn’t been a request for more than two minutes our app self starts a request

to the location services. More inaccurate locations are used after ten minutes without an more accurate position. Furthermore we only push our new location to the Sechzehn servers, when the user moved more than 100 meters.

4. Incognito Mode:
   Within his settings the user can disable that other users of Sechzehn can see his position. This feature allows privacy without logging out.
5. Live Chats:
   Our app not only gives you the opportunity to see friends on a map, you can also contact them via live chats. Those live chats also support read receipts.
6. Venue Filters:
   The venue search offers filtering for opening hours and price category.
7. Push Notifications:
   The user receives push notifications if a new message is sent to him.
8. Web Interface:
   We also provide a web interface to confirm email addresses and set new passwords.
9. Live Updates:
   The location is updated every 100m to provide a near to real-time feeling.
10. Encryption:
    We use TLS as transport encryption to prevent eavesdropping.
11. Venue Marker Colored by Rating:
    Our venue markers are colored by the rating of the venue. Where green means a higher rating and red a lower rating. The colors of other ratings are interpolated between these colors.

3.4 App Flow

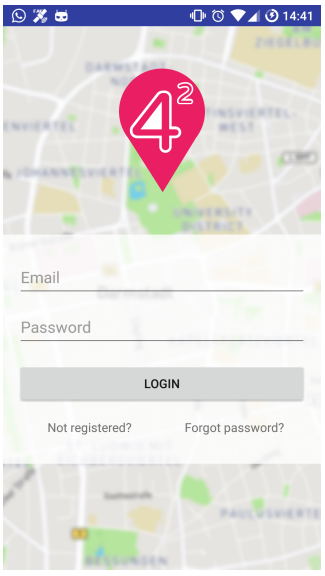

Figure 3.1: Login View

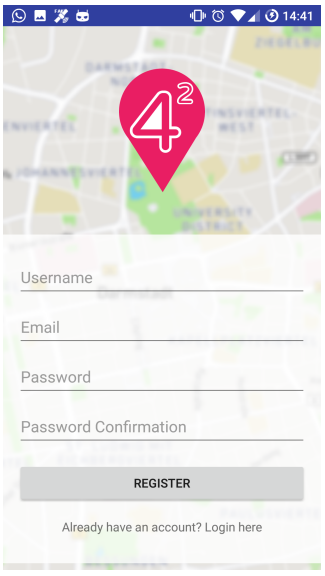

Figure 3.2: Register View

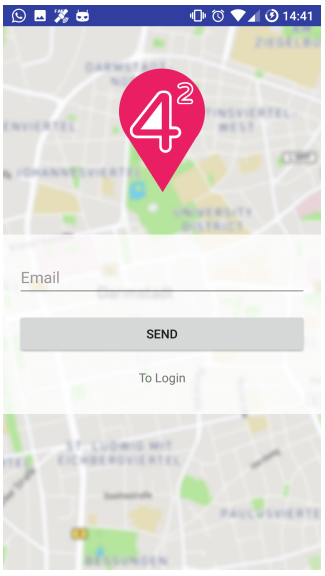

Figure 3.3: Forgot PW View

When the user starts our app for the first time, he enters the login view. This view is shown in Figure 3.1. If the user enters his login credentials correctly, he is forwarded to the map view shown in Figure 3.4. During the first start of our app, the user also has to accept the permissions mentioned in

section 3.1. Wrong credentials provided by the user lead to red warnings indicating the problem. The login credentials are stored to improve the user experience by not asking for them every time.

If the user has no account to log in, he can create a new one. To do this he has to click on the “Not registered?” button. A click on this button directs the user to the register view shown in Figure 3.2. The username as well as the mail address have to be unique. With this restriction we can ensure that there is only one account per mail address. We ensure that the password is free of typos by asking for the password two times. Only if the password is entered correctly twice, an account is created. A toast indicates a successful registration. After a successful registration, the user is redirected to the login view and a confirmation mail is sent to the given mail address. The login only works if the mail address is verified. To confirm the account a the link in the email has to be clicked. The smartphone offers the possibility to open the link via a browser or the app “Sechzehn”. If the link is opened with “Sechzehn” the account will be activated and the login screen is shown again.

We also provide the option to reset the account password if necessary (e.g. due to memory loss). The user can enter the *Forgot PW View* shown in Figure 3.3 by clicking the “Forgot password?” button. The user now can enter the email address and click the send button. A password reset request will be sent to that email address. The link in the email could be opened with “Sechzehn” or with any browser. If the link in the email is opened with “Sechzehn” a screen is opened where you have to insert a new password (two times). By clicking the confirm button the passwords are checked and if they are equal the new password is set. The same procedure could be done on a website by opening the link in a browser.

We further assume that the user is logged in. Now he sees a bottom navigation bar with three navigation options: Search, Friends and Profile. The *Search View* shows a map with venues and users, and offers a search for venues. The *Friends View* shows a list with friends, friendship requests and the last messages. The user profile displays all information we have about the user.

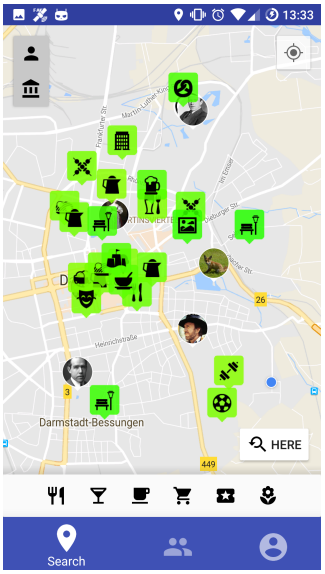

Figure 3.4: Map View

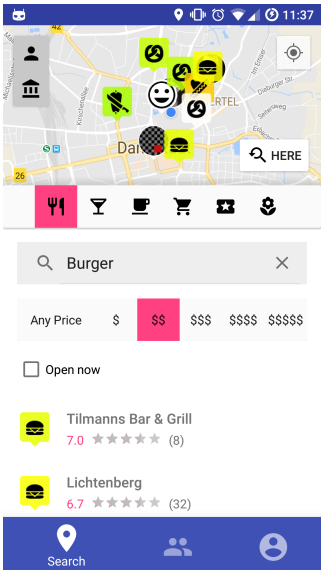

Figure 3.5: Search View

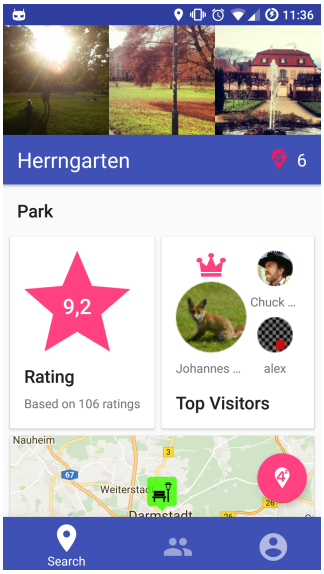

Figure 3.6: Venue View

The first view appearing after the login, is the map view shown in Figure 3.4. Per default it displays users and venues in your surrounding. Two buttons in the top left corner can be used to hide and show venues and users. The six icons right on top of the bottom navigation bar, can be used to trigger fast searches for food, drinks, coffee, shops, arts, and outdoor activities. If the map is moved around the search can be re-triggered by tapping on the floating action button appearing in the bottom right corner. Tapping on a venue opens the venue view while tapping on a user opens the profile of the user. By sliding up the quick search bar, the extensive search view shown in Figure 3.5.

To find the venue of the users interest, we offer a more extensive search. Besides a keyword search, we also offer filtering for a price category and only displaying venues that are open at the moment. The search results are displayed in the map as well as in a list right below the search query. The filtering for categories mentioned in the previous paragraph, also works in this view. The results can be sorted by rating or by distance.

The venue view is shown in Figure 3.6. The venue view summarizes information about a venue provided by the community. This includes an average rating, pictures of the venue and comments that can contain a photo. Furthermore general information about the venue is shown. With general information we mean opening hours, address, and contact details. With a floating action button it is possible to check in into a venue and rate it. Based on those check ins we provide a bit of gamification: the three top visitors of a venue are highlighted to encourage users using the check in function and thereby rate the venue.

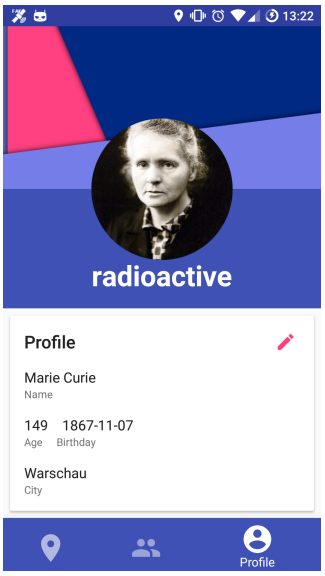

Figure 3.7: User Profile

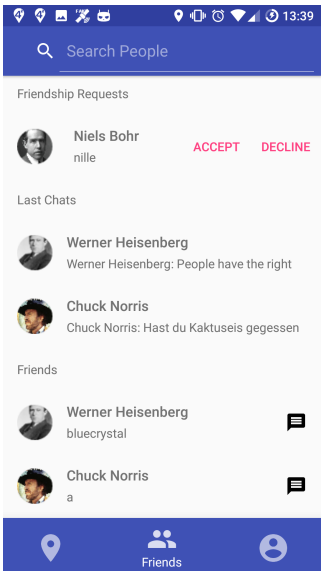

Figure 3.8: Friends View

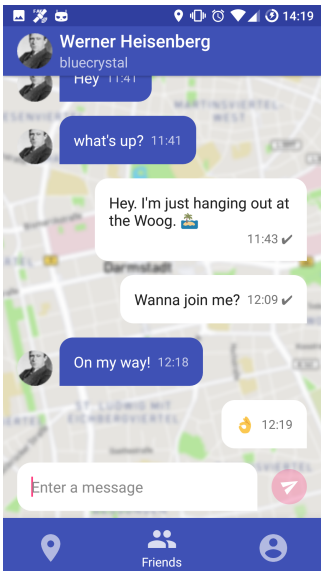

Figure 3.9: Live Chat

The profile view is shown in Figure 3.7. The profile picture is on the top of the profile view. By tapping on it, the users photo gallery is opened and he can choose a new profile picture. In the profile section basic information about the user is displayed (respectively real name, age and home town). This information can be changed by clicking on the small pencil in the top right of the section. In this edit menu you also have the option to toggle the incognito mode described in section 3.3. In this view the user can change all of this information. Furthermore this is where the log out option is located. In the bottom of the profile view there are several account settings not described in detail here.

The friends view is shown in Figure 3.8. In the first section open friendship requests are displayed. These requests can be accepted or declined. In the second section the chats are displayed. In the last section the user's friends are shown. By tapping on a friend his profile is shown, by tapping on the message symbol the live chat with the user is invoked. All of those sections are only displayed if information in them is available. Without outstanding friendship requests, the section friendship requests is not shown. It is tedious to scroll through this growing list. We therefore provided a search bar to filter the list. The search bar is also used to find and add new friends.

The chat view is shown in Figure 3.9. On the top of the chat view the chat partner is shown. With a tap on the name of the chat partner, his profile is entered. Checks marks next to the messages indicate that you chat partner has read a message of yours. To make the live chat useful even if the app is not in the foreground, we provide push notifications. This push notifications are displayed for every incoming message. They contain the name of the chat partner and the first few words of the message.

4 Feature List

- Account:
  - create an account (registering and confirming email)
  - login (with email and password)
  - password reset if password forgotten (send token and open via app/browser)
- Venue details:
  - pictures (album with swiping)
  - top visitors (three)
  - contact (address, phone, web)
  - opening hours
  - comments (add new comments (w/ photos) and rate it)
  - check in and rate the venue
- Search
  - list of venues (ordered by rating)
  - map with venues (venues coloured by rating)
  - filterable (by venue category, price category, and opened now)
- Map:
  - shows venues on the map (click to view the venue detail)
  - shows user location on the map
  - show friends location (with live updates)
- User Profile:
  - show and edit profile (edit name, birthday, city)
  - view/change profile picture
  - logout
  - delete account
  - change email
  - change password (when logged in and old password is known)
  - reset password (with email if password is not known)
- Friends:
  - find friends (by name/username, comments, nearby people (if incognito mode is disabled))
  - friend list (show friends, friendship requests, and link to profile by taping on a user)
  - chat (write messages,see last messages, and become push notification)

Bibliography

[1] *Sechzehn API Documentation | Apiary*, Accessed: 2017-08-24,
http://docs.iptk.apiary.io/

[2] *Heroku*, Accessed: 2017-08-09,
https://www.heroku.com/

[3] *Cloudinary Features*, Accessed: 2017-08-09
http://cloudinary.com/features

[4] *API Reference Documentation*, Accessed: 2017-08-09,
https://nodejs.org/en/docs/

[5] *AdonisJs at a Glance*, Accessed: 2017-08-09,
https://adonisjs.com/docs/3.2/overview

[6] *PostgreSQL: Documentation*, Accessed: 2017-08-09,
https://www.postgresql.org/docs/

[7] *Introduction to JSON Web Tokens*, Accessed: 2017-08-09,
https://jwt.io/introduction/

[8] *Data Binding Library*, Accessed: 2017-08-09,
https://developer.android.com/topic/libraries/data-binding/index.html

[9] *Retrofit: A type-safe HTTP client for Android and Java*, Accessed: 2017-08-09,
https://square.github.io/retrofit/

[10] *Google GSON*, Accessed: 2017-08-09,
https://github.com/google/gson

[11] *ChatKit for Android*, Accessed: 2017-08-09,
https://github.com/stfalcon-studio/ChatKit