Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Tim Staats

Extending an Android App for a Multi-user Geodatabase of Historical Monuments

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Tim Staats

Extending an Android App for a Multi-user Geodatabase of Historical Monuments

Bachelorthesis based on the study regulations
for the Bachelor of Engineering degree programme
Informations- und Elektrotechnik
at the Department of Information and Electrical Engineering
of the Faculty of Engineering and Computer Science
of the Hamburg University of Applied Sciences

Supervising examiner : Prof. Dr. rer. nat. Henning Dierks
Second Examiner : Prof. Dr.-Ing. Marc Hensel

Day of delivery 20. Mai 2019

Tim Staats

Title of the Bachelorthesis

Extending an Android App for a Multi-user Geodatabase of Historical Monuments

Keywords

Java, Android, Webapplication, Backendless, Geodata, Userservice, historical monuments, google map, Baas

Abstract

This thesis documents the development of an Android Application. The app offers the opportunity to registered users to capture and catalog historical Monuments. Furthermore the app displays the data within a google map and in form of a detailed list. The corresponding Backend manages the data and also display it within a webapplication. The necessary measures, concepts, problems and their solutions are described below.

Tim Staats

Titel der Arbeit

Weiterentwicklung einer Geodatenbasierten Multi-Nutzer Android App für historische Monumente

Stichworte

Java, Android, Webapplication, Backendless, Geodaten, Nutzerservice, historische Monumente, Google Map, BaaS

Kurzzusammenfassung

Diese Arbeit dokumentiert die Weiterentwicklung einer Android Applikation. Die App bietet registrierten Nutzern die Möglichkeit historische Monumente zu erfassen und zu katalogisieren. Des Weiteren visualisiert die App die katalogisierten Daten auf eine google Map und in Form von detailierten Listen. Im dazugehörigen Backend werden die Daten verwaltet und in einer Web Applikation visualisiert. Die dafür erforderlichen Maßnahmen, Konzepte, Probleme und deren Lösungen werden im folgenden beschrieben.

Contents

List of Figures

Listings

1 Introduction

This bachelor thesis describes the development process of a native android application. Today life without smart-phone has become unimaginable. It has infiltrated nearly every possible part of humans society. *In the year 2018 two out of three of the people worldwide and 81% in Germany owns a smart-phone.*

Applications look like a technical promise of the future. But reminiscending the past is very important to the mankind. Like Elie Wiesel once said: "*Without memory, there is no culture. Without memory, there would be no civilization, no society, no future.* - Gefunden auf: https://www.myzitate.de/erinnerungen/Without memory, there is no culture. Without memory, there would be no civilization, no society, no future - Gefunden auf: https://www.myzitate.de/erinnerungen/".

Therefor the Civitas app should help to keep the memory, and store it for upcoming generations. In detail, it is an approach to store historical monument of the roman empire.

1.1 Previous team achivement

- User registration
- User rights plus Admin
- Filtering
- Create artefacts
- Edit artefacts

1.2 Problem statement or Method

no manual to setup the given project to continue and improve the functionality. on the other hand, use the gained expirience with BaaS to provide a Android and Web app with nice features, such as google map, forget password, mailing service for free. Decision to make: spent time to get it running but dont know where to start or start from scratch and provide an app with nice features.

1.3 Criteria for success or objective

everthing is working fine. no bugs around

1.4 Backendless

BaaS Back-end as a service.

1.5 Java

Object orientated programming

Figure 1.1: Girl with ballon[1]

ich führe ein...

[1] URL: https://www.banksy.org [cited 22 August 2018]

2 Analysis

2.1 images

glide vs universal

2.2 backend

REST vs backendless vs firebase

handle response() handle fault()

2.3 user service

Backendless provides user registration, login, password recovery, message service

2.4 webapplication

user data, data storage, file storage, geolocation api

2.5 usability

ich analysiere!

3 Design

3.1 Navigation in general

3.1.1 Action Bar

- Search, filter
- Location
- Edit artefact
- Delete artefact

3.1.2 Navigation Drawer

- Map
- Artefacts
- Impressum
- Settings
- Logout

3.2 Login

3.3 Registration

3.4 map

artefacts represented by custom marker. display first info glimpse within snippet. info contains name and description, custom marker represents category.

3.5 list

all available artefacts displayed in recyclerview with image, name, category.

3.6 detail

accessable from map marker click or list item click. provides full infos

3.7 filter

filtering artefacts list via lifesearch with name, or category or age and display result within list and also on map and vice versa

Difficulties: keep the filter result through various functions, like filtered List to artefact Detail and back to filtered List. Or the same on the map and returning after marker click

3.8 edit

3.9 Impressum

4 Realization

Beschreibt den Prozess des programmierens der App, die Probleme dabei und die dazugehÃ¶rigen LÃ¶sungen.

4.1 Android

4.1.1 Application Class

An Application Class is called before the MainActivity is called for the first time. Due to the usage of Backendless (BaaS) the applicationId, Api Key and the serverUrl are stored in the ApplicationClass and connects the Webapplication with the Android Application. It also contains variables for the user credentials saved in the Backendless.user table and the Artefact table.

4.1.2 Backendless

Backendless provides the server data to the app when it is requested. This happens in case of registration, login, artefact creation, file upload, file download, geolocating marker, edit artefact, etc. Independently which data is needed, an asynchronous Backendless request requires the implemention of two interface methods. The asyncCallback methods "handleResponse()" and "handleFault()" are recommended to keep the main thread free and increasing the application performance. "handleFault()" is called if something went wrong. "handleResponse()" is called if the request was successful and provides the requested data.

User table

With Backendless one can create data tables with ease. The user table should contain certain properties such as name, email and password. To get a table like that one calls the Backendless.UserService.register() method and pass a BackendlessUser object as argument.

An exemplary interaction between app and the Backendless server is displayed here in case of user registration.

```
1  BackendlessUser user = new BackendlessUser();
2  user.setEmail(email);
3  user.setPassword(password);
4  // for adding columns to backendless, use setProperty like below
5  user.setProperty("name", name);
6
7  Backendless.UserService.register(user, new AsyncCallback<BackendlessUser>() {
8          @Override
9          public void handleResponse(BackendlessUser response) {
10
11                 showProgress(false);
12                 mainActivity.fragmentSwitch(new LoginFragment(), false, "");
13
14         }
15
16         @Override
17         public void handleFault(BackendlessFault fault) {
18                 Toast.makeText(getContext(),
19                                getResources().getText(R.string.toast_backendless_error)
20                                + fault.getMessage(),
21                                Toast.LENGTH_SHORT).show();
22                 showProgress(false);
23         }
24 });
```

Listing 4.1: Backendless user registration

The method handleResponse() returns to the LoginFragment, waiting for the user to login. From now on the user table is listed within the Backendless console (Webapplication) 4.5.

Artefact table

contains artefact data

Figure 4.1: Artefact Table[1]

4.1.3 MainActivity

The MainActivity is the starting point of this application. It contains two Fragments, Login and Register.

[1] URL: https://www.banksy.org [cited 22 August 2018]

Figure 4.2: Activity flow of MainActivity[2]

LoginFragment

This Fragment is the first view which appears to the user. It provides three possibilities.

- Login
- Register
- Password recovery

The first step for a new user is moving to the RegisterFragment via button click. If the registration is accomplished, one can login or reset password. Successful login leads to the Map,

[2]URL: https://www.banksy.org [cited 22 August 2018]

which is the main part of the application. It keeps the user logged in until the logout button is clicked, hence the next time Civitas is started it navigates directly to the Map.

The Login progress is displayed here.

```
1  Backendless.UserService.login(email, password, new AsyncCallback<BackendlessUser>() {
2          @Override
3          public void handleResponse(BackendlessUser response) {
4                  ApplicationClass.user = response;
5                  Toast.makeText(getContext(),
6                                  getResources().getText(R.string.toast_login_successful),
7                                  Toast.LENGTH_SHORT).show();
8                  // if user is successful logged in
9                  retrieveArtefactsFromBackendless();
10                 //navigateToMapActivity();
11         }
12
13         @Override
14         public void handleFault(BackendlessFault fault) {
15                 Toast.makeText(getContext(),
16                                 getResources().getText(R.string.toast_backendless_error)
17                                 + fault.getMessage(),
18                                 Toast.LENGTH_SHORT).show();
19                 showProgress(false);
20         }
21 }, true);
```

Listing 4.2: Backendless user login

A basic Backendless request implements two methods, handleResponse() and handleFault(). If request is successful, the response contains the user data. This data is stored in the Applicationclass.user for further purpose.

RegisterFragment

4.1.4 Map

Figure 4.3: Activity flow of MapActivity[3]

4.1.5 ArtefactActivity

Figure 4.4: Activity flow of ArtefactActivity[4]

[3]URL: https://www.banksy.org [cited 22 August 2018]

ListFragment

ArtefactDetailFragment

NewArtefactFragment

EditArtefactFragment

4.1.6 ImpressumActivity

4.2 Web Application - Backendless Console

4.2.1 User's

email templates

4.2.2 Data

User table

The user table contains the same properties like the BackendlessUser object 4.1 where it is created from plus the properties *created* and *updated* which where provided automatically from the Backendless.

[4] URL: https://www.banksy.org [cited 22 August 2018]

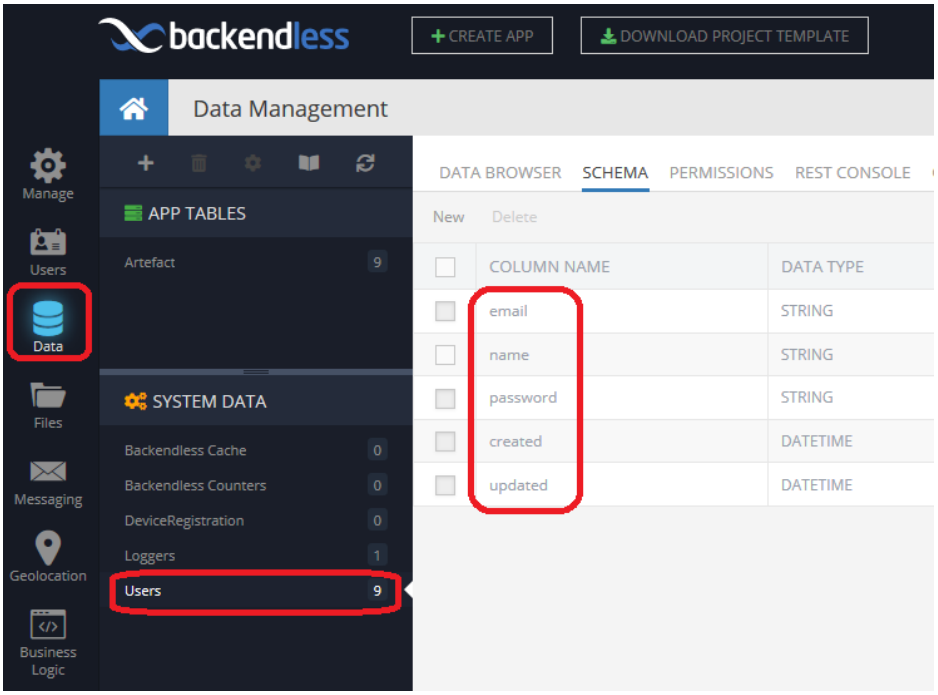

Figure 4.5: Backendless console user table

Artefact table

contains app Table contains system data

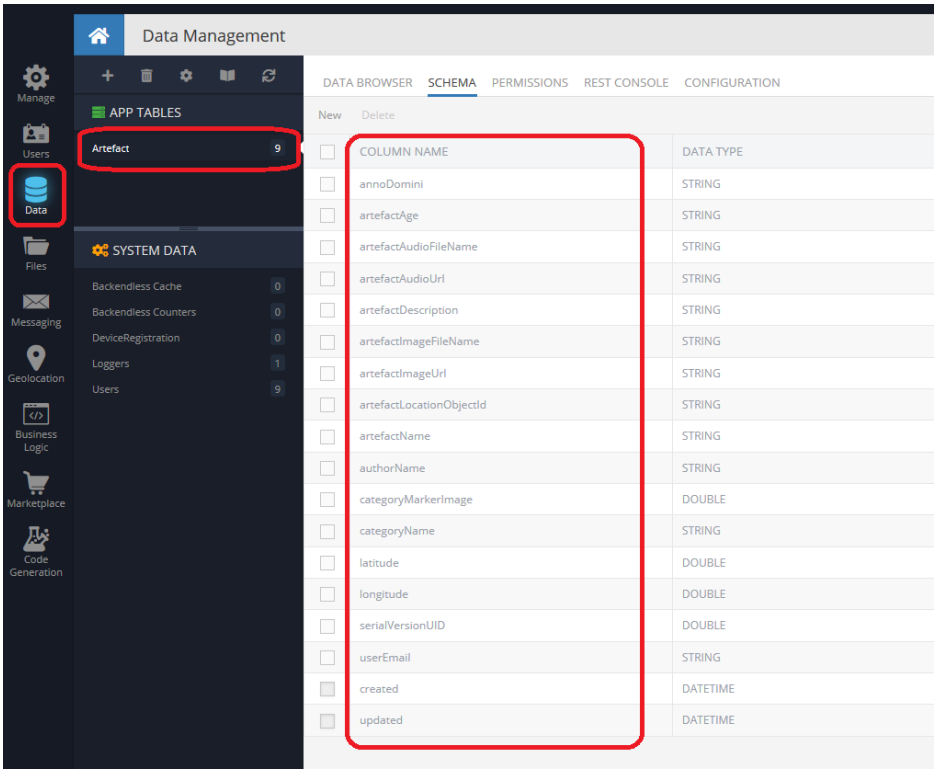

Figure 4.6: Backendless console artefact table

4.2.3 Files

images and audios

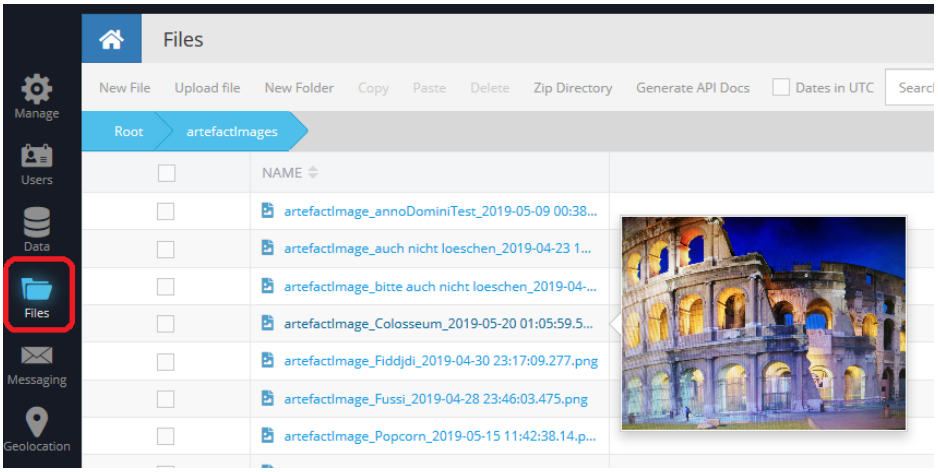

Figure 4.7: Backendless console files folder

4.2.4 Geolocation

displays artefact coordinates

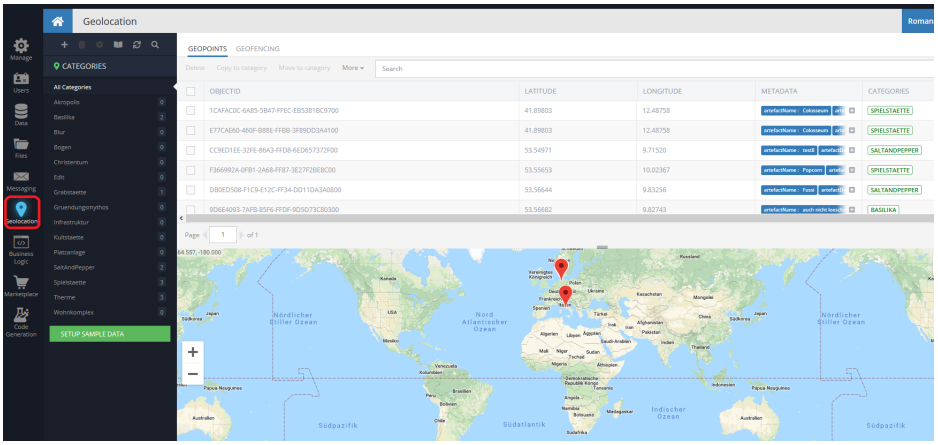

Figure 4.8: Backendless console geolocation folder

ich realisiere!

5 Tests

5.1 JUnit

- test ApplicationClass.mArtefactList
- test CategoryList

5.2 Integration Test

buttons work well ich Teste

6 Conclusion

ich fasse zusammen

6.1 Furthermore

On the one hand there is a working app with a lot of functions such as.

- Navigation through the app
- Screen returns to last focus
- Webapplication for administrational purpose (admin control panel)
- Password recovery
- List overview
- Complex filter system within list and map
- Create artefacts (everybody)
- Edit artefacts (authorized users)
- Delete artefacts (authorized users)

But on the flip side of things, there is still a lot of work to do.

- MP3
- Share
- Rating system
- Thumbnail within marker info window
- Edit user credentials
- Multiple image related to a single artefact
- Comment function for artefacts

- Retrieve artefacts in relation to the device location

Declaration

I declare within the meaning of section 25(4) of the Ex-amination and Study Regulations of the International De-gree Course Information Engineering that: this Bachelor report has been completed by myself inde-pendently without outside help and only the defined sources and study aids were used. Sections that reflect the thoughts or works of others are made known through the definition of sources.

Hamburg, May 20, 2019

City, Date

sign