ComeTogether - an android app to meet new people

Christof Ochmann
University of Applied Sciences Zittau/Görlitz
02826 Görlitz, Germany
sichochm@hs-zigr.de

Ingo Körner
University of Applied Sciences Zittau/Görlitz
02826 Görlitz, Germany
siinkoer@hs-zigr.de

ABSTRACT

This work based on the document "Come-Together-App", which was created in Wirtschaftsinformatik II in a course of studies of tourism. The ideas of the project "Come-Together-App" are realized as a prototype in "ComeTogether - an android app to meet new people". Both, frontend and backend are analysed, desgined and implemented.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Design, Performance, Theory

Keywords

Parallel evolutionary algorithms, island model, spatial structures, offspring populations, runtime analysis

1. INTRODUCTION

In this project a prototyp named ComeTogether will be created. He runs on smartphones with the android operating system. ComeTogether is a mixture of an eventcalendar and a platonic touch. ComeTogether is an app to establish social contacts. It is not a flirt app or a dating app, but an app to meet immediately a whole group of new people simultaneously and have fun with them. The core of the application consists of an offer function and a search function. With offer you can offer events with a participating group of people. With search you can find a group of people which participate in the event you have searched. Both, users who offer and users who search signalise with an offer or a search, that they want actively meet new people.

2. PREVIOUS WORK

There are flirt apps for smartphones like "myamio - Die Partnersuche für das iPhone" which are used by singles to contact others. Beside flirt apps there are also calendar of events as app-download for enterprising people. A calender of event app is "PRINZ iPhone-App für Veranstaltungen, Bars und Clubs in Ihrer Stadt". This app is for people who mainly want to find a new locality or enterprise, independent of the people who take part. A certain link of calendar of events and flirt app provides the free "Barcardi Togethering App". In this app there is only the possibility to date your own facebook friends but you cannot meet strangers. In contrast to Barcardi app there is no need to join a social network to run ComeTogether app. What can you do, if you are not interested in relationship and your friends have no time for you but you want to go out and meet some people or you are alone in an unfamiliar environment and want to have an enterprise? The ComeTogether app is a new and modern way to contact people. You can meet new people, share common interests and promote sociability. ComeTogether can reduce barriers of single or lonely people which long for new people. Moreover, the probability of a disappointment is less if you meet a person with same interests electronically first than you have a face-to-face encounter immediately. And the electronically way has a certain distance to the unknown people and that gives more security. ComeTogether is a completely new mixture of calender of events and dating app. ComeTogether is not limited to a niche like dating apps, which are only made for singles.

3. REQUIREMENTS ENGINEERING

Figure 1 shows the use case diagram for ComeTogether.

3.1 use cases

UC 1.1 register - create profile

1. start app
2. touch button "create profile"
3. input of personal data (name, gender, data of birth, email)
4. may upload a photo
5. input password
6. repeat password
7. touch button "register"

UC 1.2 login

1. start app
2. user is automatically logged

UC 2.1 create event

1. UC 1.2 login
2. touch "Offer" - Button
3. input data (date, place, titel) on the offer form
4. touch button "okay"
5. the generated event is added to the database and displayed on the search page

UC 2.2 search events

1. UC 1.2 login
2. touch "search" - button
3. display all events in table form location based (via GPS in a 50km radius): à 3 categories (events, food & beverage, sports- and leisure activities in table form):
   (a) date
   (b) place
   (c) title of event
   (d) number of participants and gender

   (a) alternative scenario: input of specific date, any other place with different radius, category thru scrolling screen right to automatically change and issuance of the new results, according to the data mentioned
4. select desired event thru touch of a titel
5. display event description and with scrolling provider profil
   (a) alternative scenario 1: select an event thru touch of the "get in touch" button
      i. open the mailbox with empty textbox and registered recipient or tenderer thru touch of the "get in touch" - button
      ii. writing a message and come into conatct with the provider thru sending the message
      iii. select number of participants and gender
   (b) alternative sceanario 2: return to the overview thru back-button to select another event

UC 2.3 delete event

1. UC 1.2 login
2. touch "my events", to display a list of own events
3. select the own event in the list
4. delete event thru touch the delete button and the system generates automatically a denial mail for persons who already participate

UC 2.4 delete past events after a day

1. past events are deleted automatically by the system after one day

UC 2.5 Event update / predict seekers

1. UC 3.2 message recall
2. tenderer touches commitment button to update number of participants and their gender

UC 2.6 cancel an event

1. UC 3.1 message recall
2. bidder touches cancel button, to generates automatically a denial mail for persons who already participate

UC 3.1 message recall

1. UC 1.2 login
2. touch mailbox button
3. select message

UC 3.2 write message

1. UC 3.1 message recall
2. touch reply button
3. write message
4. touch send button

UC 3.3 notify user if a new message arrives

1. system sends automatically an email to the user

4. DESIGN

In figure 2 you can see the design for the UserService. UserServiceREST encapsulate the REST-Functionality of the UserService. UserPersistence access database with prepared statements p.e. to save, read or delete users. Between UserServiceREST and UserPersistence are service class and DAO class to encapsulate different levels of abstraction. Beside UserServiceREST there are the classes EventServiceREST, ParticipationServiceREST and MessageServiceREST. EventServiceREST in figure 3 creates, reads and deletes events. ParticipationServiceREST in figure 5 creates participations and gives back a list of participation for a given eventid or a given userid. MessageServiceREST in figure 4 creates, reads or deletes messages.

5. PROGRAMMING ENVIRONMENT

- Eclipse 3.7.2
- Git 1.7.11.2
- github.com
- Google Plugin for Eclipse 2.5.2
- ADT plugin for eclipse 16.0.1
- m2eclipse plugin 1.0.100
- JDK 1.7
- Tomcat 7.0

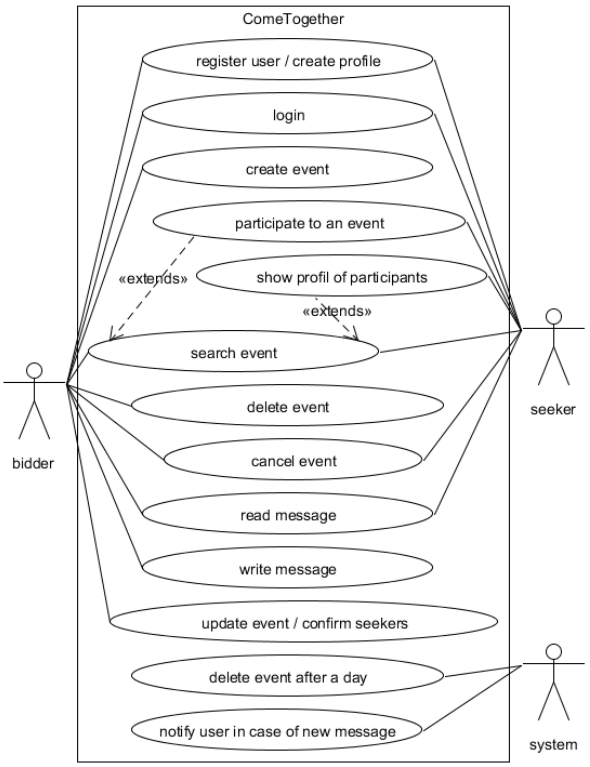

Figure 1: use case diagram

- JUnit 4.8.2
- Maven 3
- Google Guice 3
- UMLet 11.3
- EasyMock 3.0
- JBoss Resteasy 2.2.1
- Jackson 1.9.2
- apache http 3.1
- PostgreSQL 9.1.4

6. BACKEND

User, events, messages and participations are stored on the server side, cause they are to many to store them all locally and keep them up to date. That is to say ComeTogether needs an internet connection to get access to all the data. A convenient way to save data remotely is provided by Platform as a Service (PaaS). With PaaS no own infrastructure is needed, because extern infrastructure is used. With PaaS the developer does not need to worry about patches, backups and scalability. Load peaks are intercepted and rapidly growing data is absorbed. PaaS is based on Infrastructure as a Service (IaaA). Computing power for running applications and memory for storing data is provided thru IaaA. For instance amazon provides computer power with EC2 and memory with S3. Often with PaaS

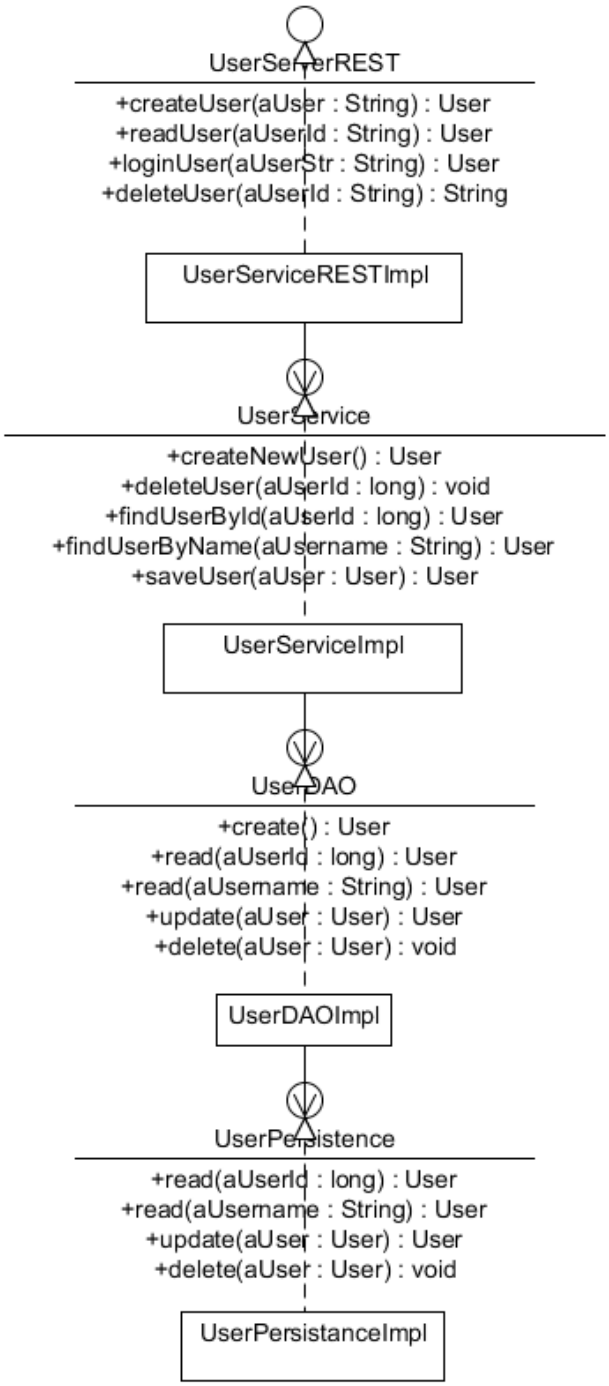

Figure 2: desing class diagramm UserServiceREST

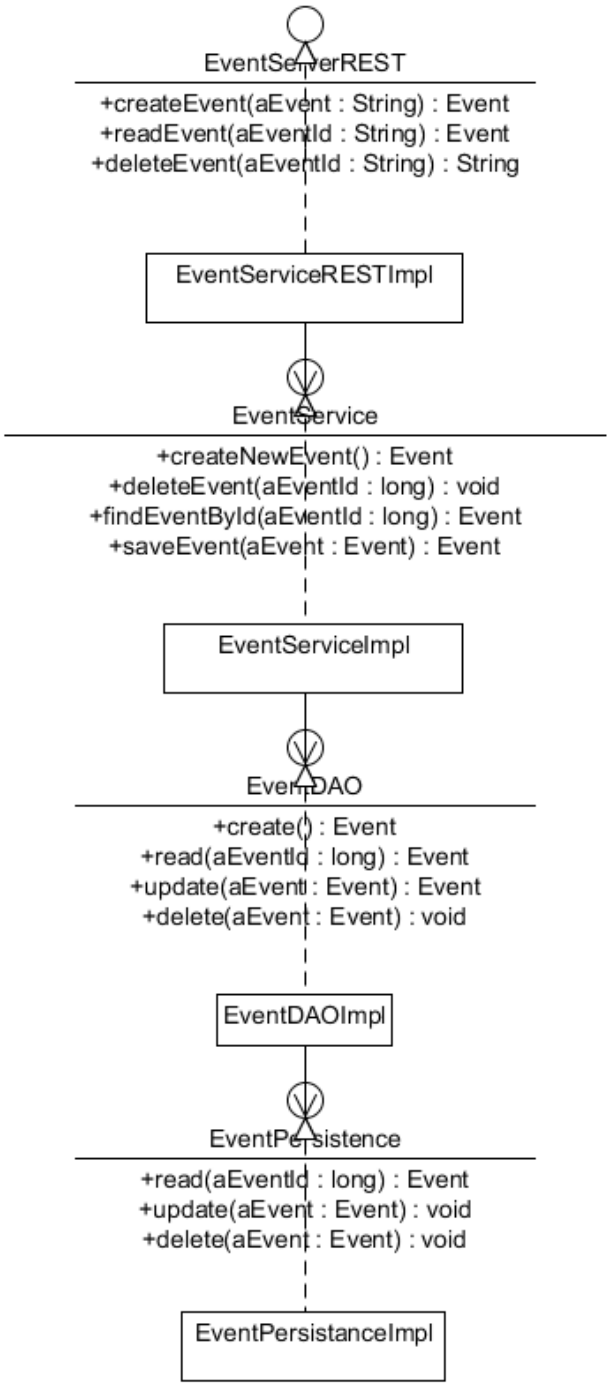

Figure 3: desing class diagramm EventServiceREST

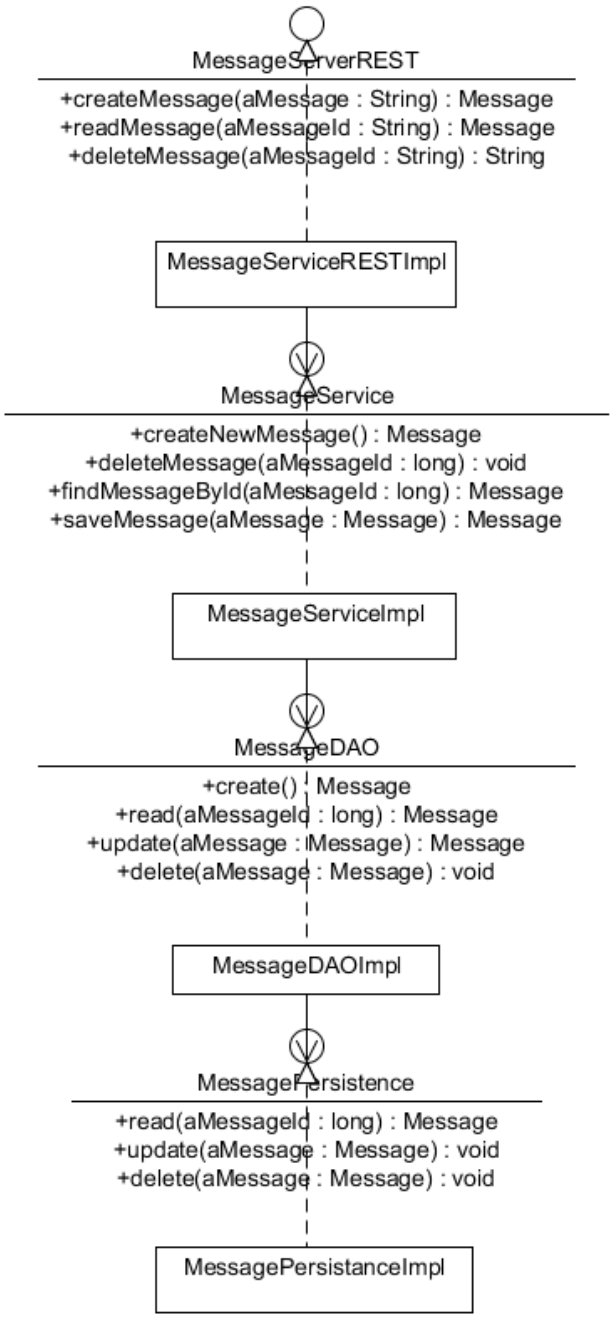

Figure 4: desing class diagramm MessageServiceREST

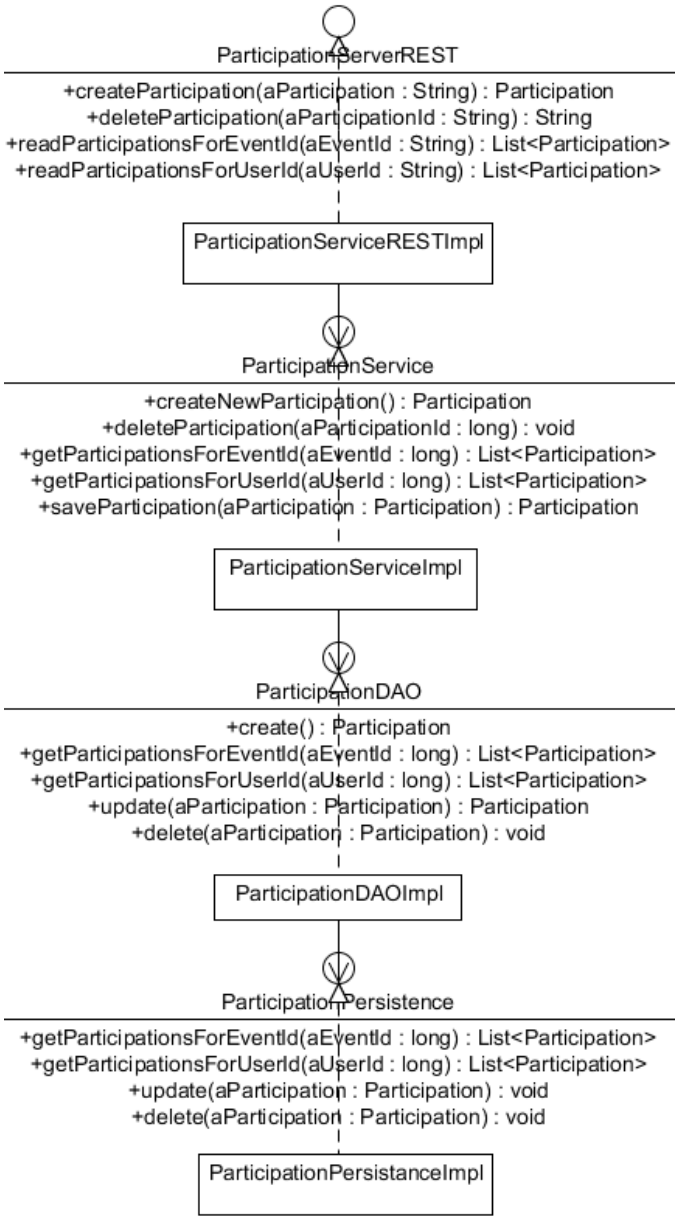

Figure 5: desing class diagramm ParticipationServiceREST

the developer has also access to a data base. Further more PaaS provides a own runtime environment as a service for which the developer paid on demand. Examples for PaaS are Google App Engine, Windows Azure or Amazon Elastic Beanstalk. GAE is usefull because google and android are closely interlinked. Amazon elastic beanstalk and microsoft windows azure are not discussed due to time and so only GAE is considered. GAE provides a JavaVM for business logic, where the backend of ComeTogether will be running. With the "datastore" a transaction save no-sql data base management system based on Googles "Big Table" concept is provided for storing data of ComeTogether. For Java also parts of JPA are supported. There are plugins like google plugin for eclipse which support developers when creating android apps with GAE. Normal communication with GAE is done via REST-Services. In GAE a servlet is running which gets http requests, does a db access and return the data via http response. Over "cloud to device messaging" a device will be notified if there is e.g. a new event. Unfortunately, the team have a lack of experience to deal with all the error messages which have occurred already in the sample project on the manufacturing side https://developers.google.com/eclipse/docs/creating_new_webapp. One error messages was "C:/Users/Ingo/android-sdks/tools/lib/proguard.cfg (Das System kann die angegebene Datei nicht finden)". The problem was described on several web pages and could be solved. An other error message was "Setup could not finish - Unable to open connection to server." It was solved by integrating the Google API in AVD for C2DM instead of the android SDK. There was an other error message while running the sample project, which could not be solved due to time: "Could not find method com.google.web.bindery.requestfactory.vm.RequestFactorySource.create, referenced from method de.ct.Util.getRequestFactory" So there was only the conventional way to host the app on a dedicated server.

7. DATABASE

In figure 6 you can see the data base design of ComeTogether.

The data is written into the data base tables through prepared statements. Prepared statements contain placeholders for the actual data to be written. Prepared statements are a fast choice if the statements differ only in the parameter values.

8. MISCELLANEOUS

To test the backend classes EasyMock 3.0 is used. EasyMock is a test framework for the dynamic generation of mock objects for interfaces and classes. Mock objects are used to unit testing java classes.

The backend uses the architecture pattern dependency injection to minimize dependencies between objects. With dependency injection, the object does not creates its dependent objects anymore. These dependencies are created by a framework. The code of the object become independent of its environment. In this way the object is easy to test because dependencies are available centralized. The backend uses the dependency injection framework called google guice.

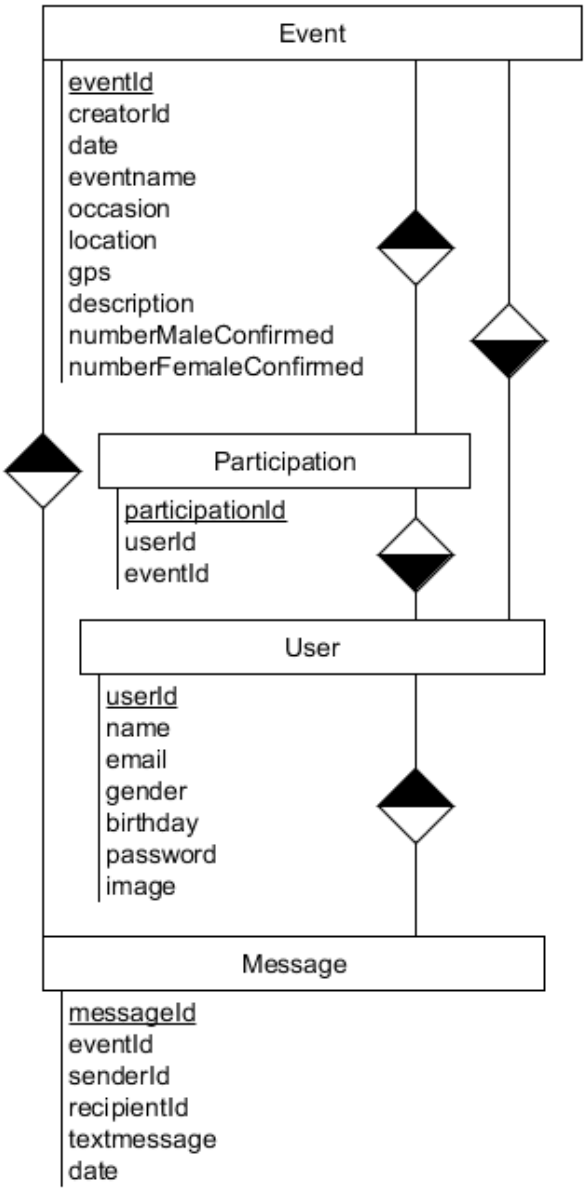

Figure 6: eer-diagram

For marshalling objects and unmarshalling JSON strings Jackson is used. Jackson is a multi-purpose Java library for processing JSON.

The communication between client and backend is done by apache http. The client uses post-requests.

9. CONCLUSION AND FUTURE WORK

For this project a lot of time was needed only to create a prototype of ComeTogether. It was time consuming to find the proper way to implement the communication between the android client and the backend. And there was a lack of knowledge to handle the errors while creating this project on google app engine. In a future work new features like a picture upload or a radius search could be implemented and the backend could be moved to a Platform as a Service provider.

10. PRELIMINARIES

11. PREVIOUS WORK

12. SORTING

...

13. SHORTEST PATHS

...

14. EULERIAN CYCLES

...

14.1 Edge Walks

...

14.2 Restricted Mutation Operators

...

14.3 Adjacency List Matchings

...

15. CONCLUSIONS

Acknowledgments

The authors would like to thank .....the German fast food industry for keeping us alive.

16. REFERENCES