

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, designed 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

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 calendar 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 calendar 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.

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

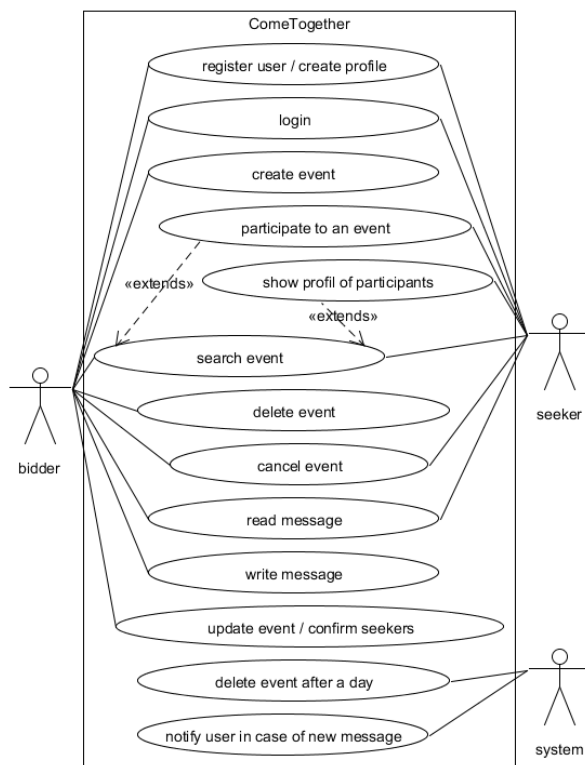


Figure 1: use case diagram

- 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
- 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

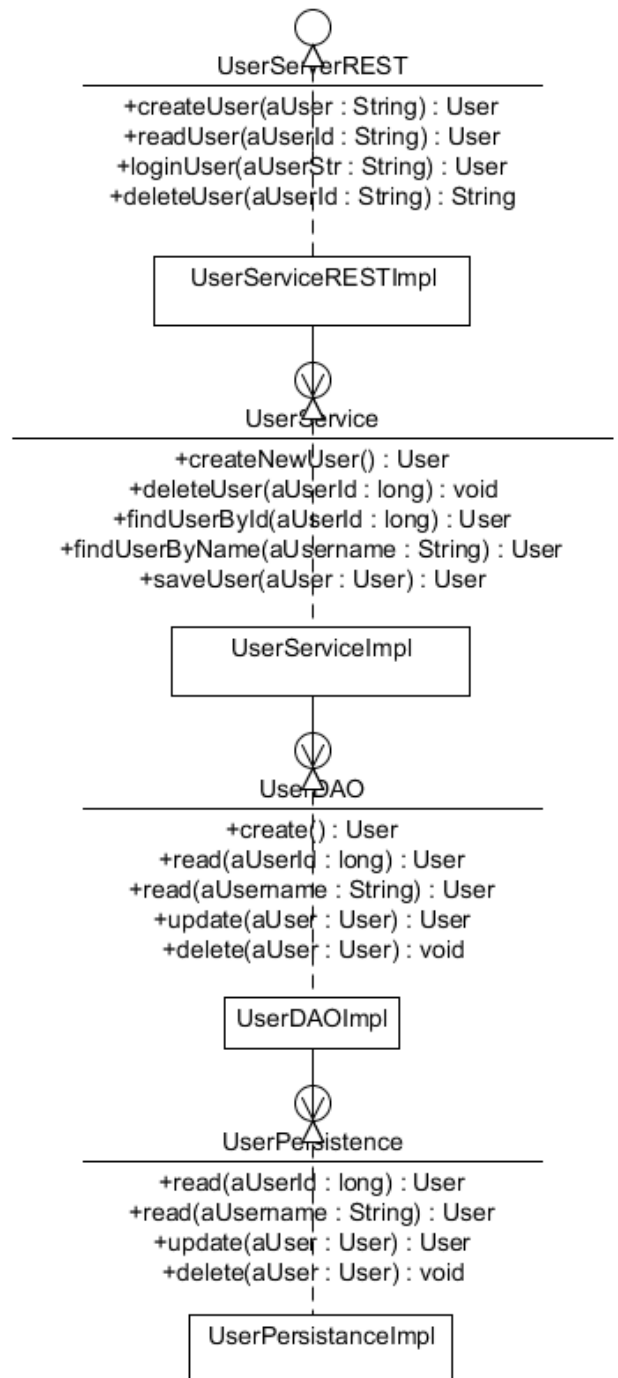


Figure 2: desing class diagramm UserServiceREST

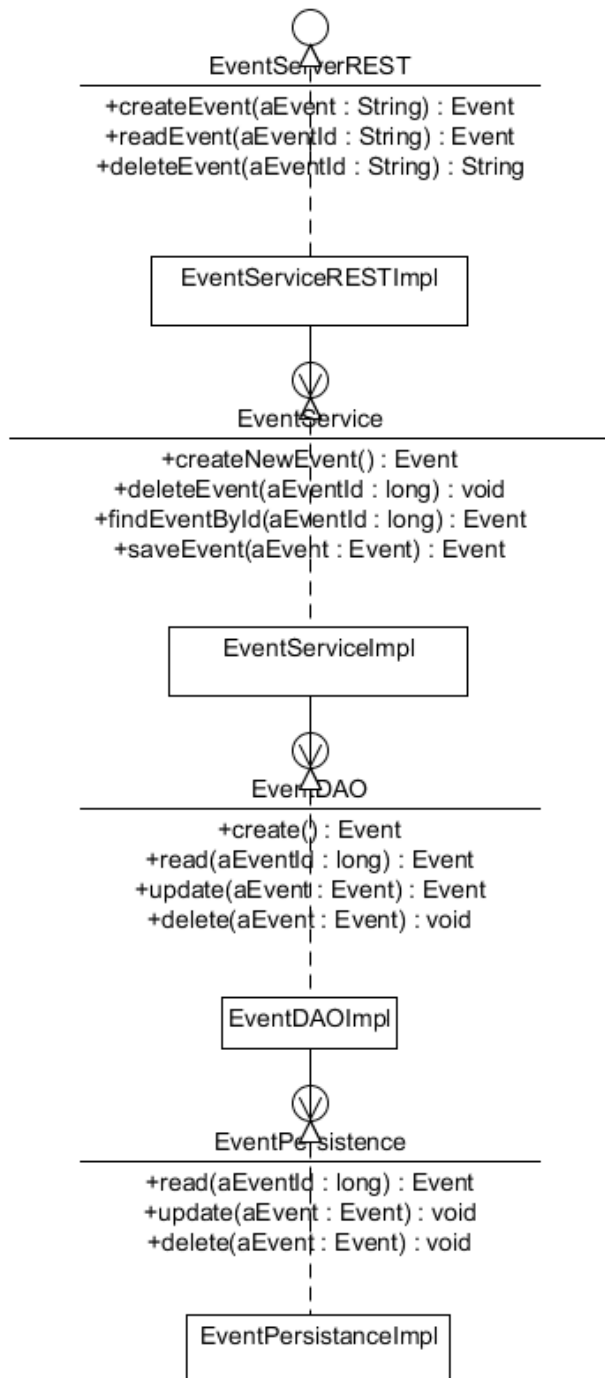


Figure 3: desing class diagramm EventServiceREST

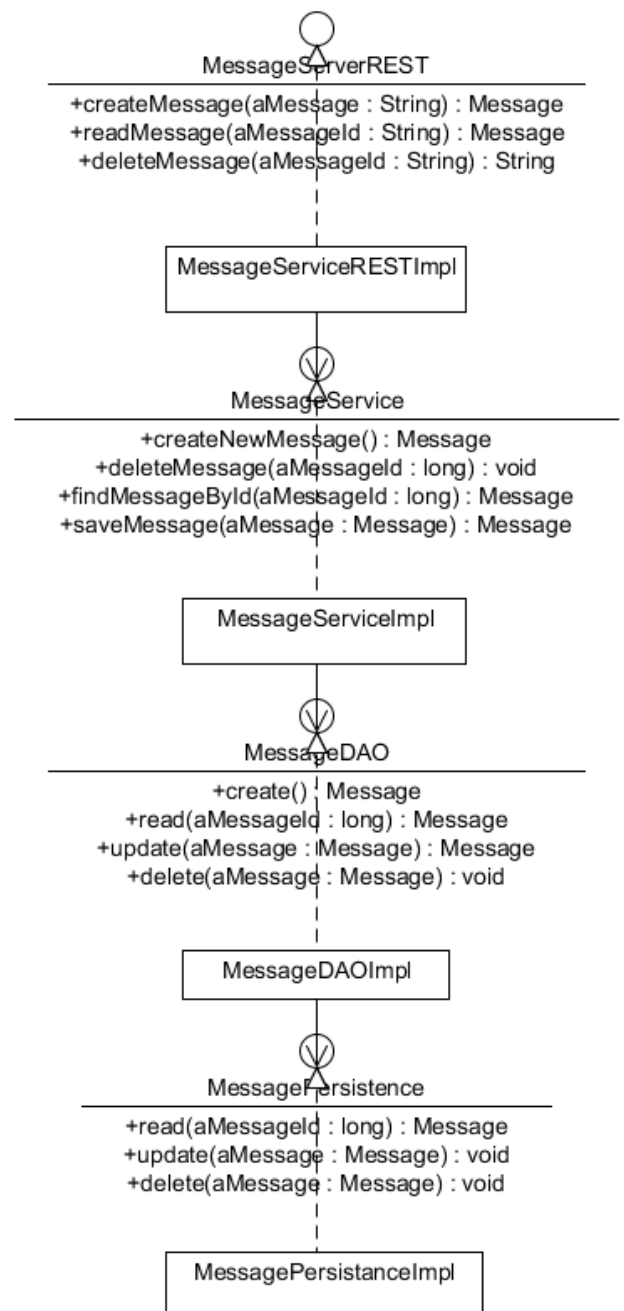


Figure 4: desing class diagramm MessageServiceR-EST

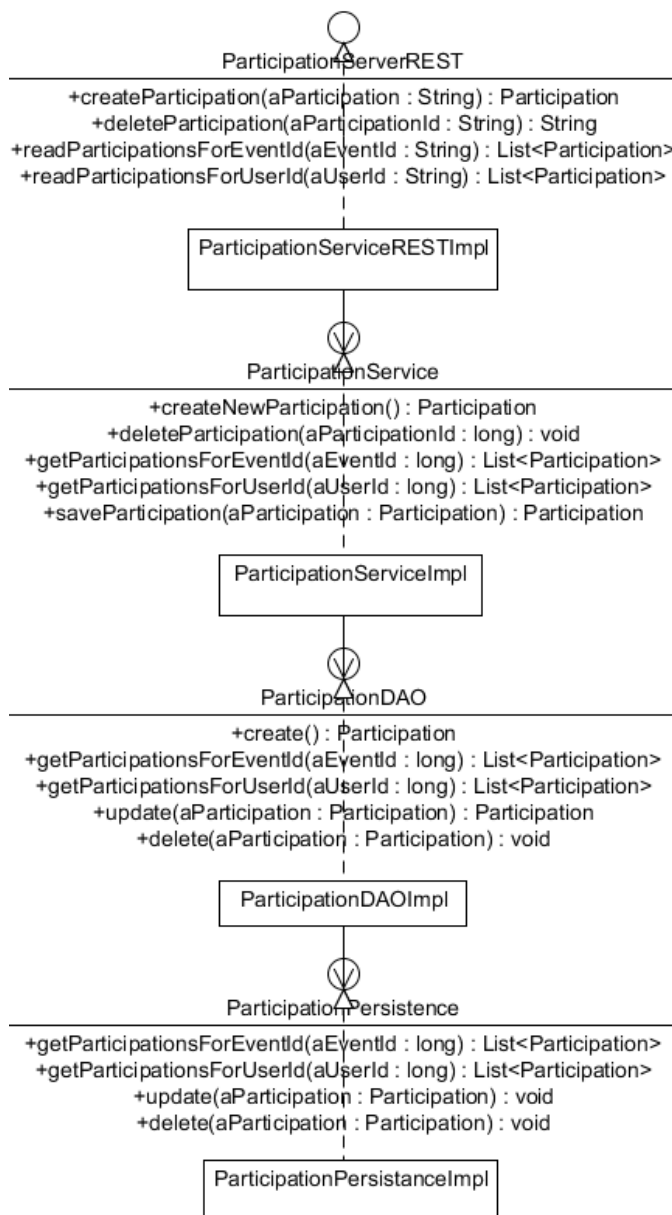


Figure 5: desing class diagramm ParticipationServiceREST

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 (IaaS). Computing power for running applications and memory for storing data is provided thru IaaS. For instance amazon provides computer power with EC2 and memory with S3. Often with PaaS 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.

8. PRELIMINARIES

9. PREVIOUS WORK

10. SORTING

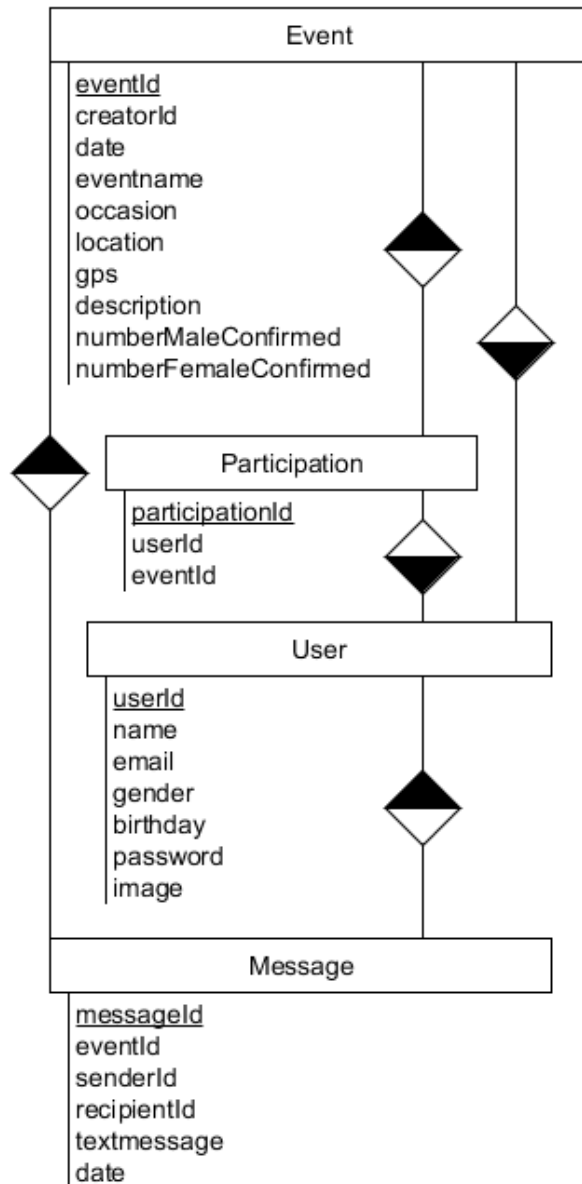


Figure 6: eer-diagram

11. SHORTEST PATHS

12. EULERIAN CYCLES

12.1 Edge Walks

12.2 Restricted Mutation Operators

12.3 Adjacency List Matchings

13. CONCLUSIONS

Acknowledgments

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

14. REFERENCES