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.

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

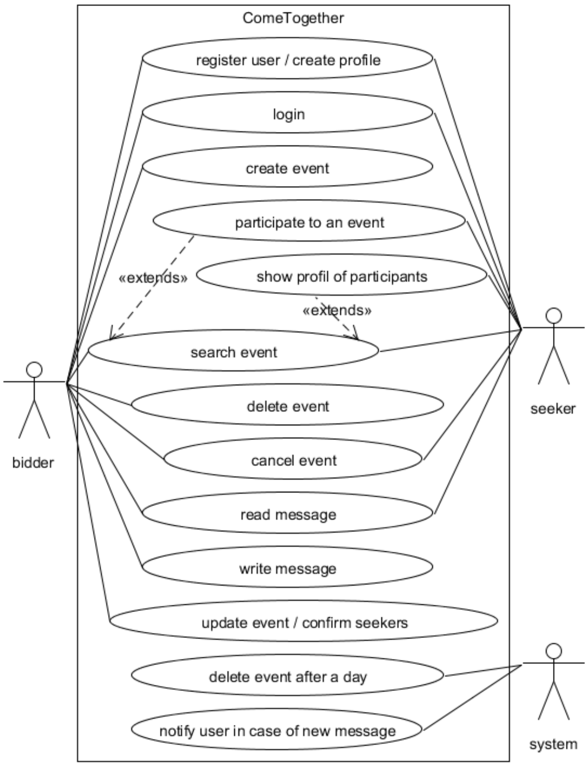

Figure 1: use case diagram

- 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

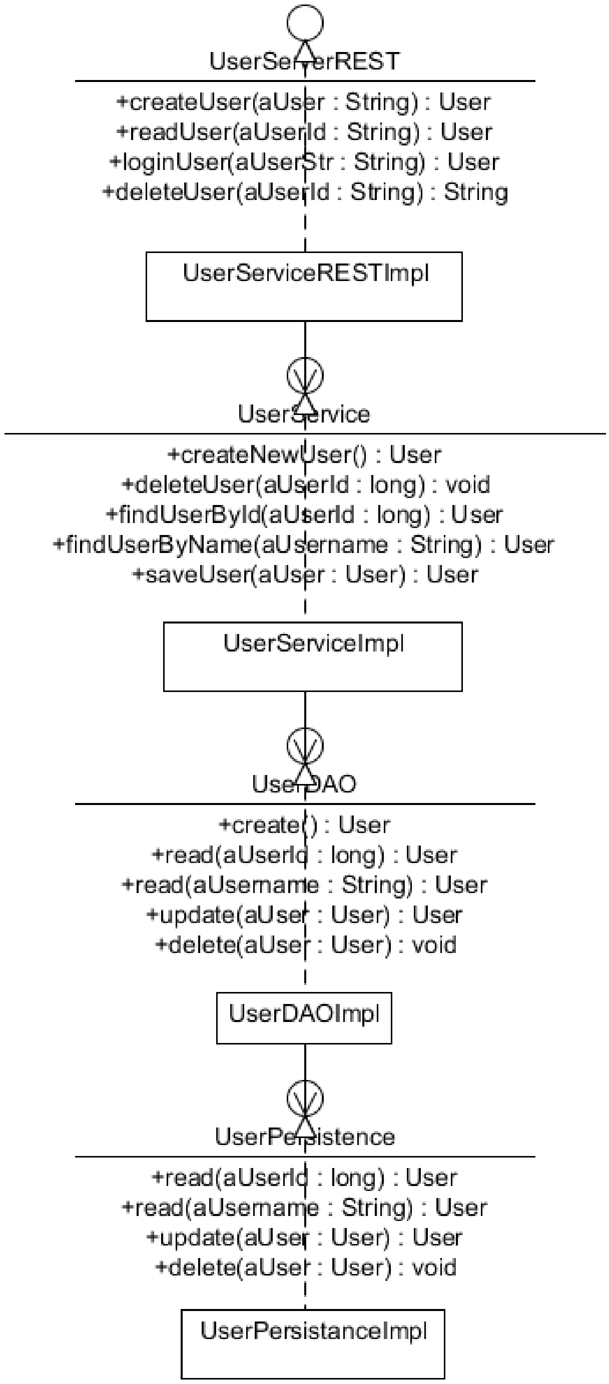
UserServerREST
+createUser(aUser : String) : User
+readUser(aUserId : String) : User
+loginUser(aUserStr : String) : User
+deleteUser(aUserId : String) : String
UserServiceRESTImpl
UserService
+createNewUser() : User
+deleteUser(aUserId : long) : void
+findUserById(aUserId : long) : User
+findUserByName(aUsername : String) : User
+saveUser(aUser : User) : User
UserServiceImpl
UserDAO
+create() : User
+read(aUserId : long) : User
+read(aUsername : String) : User
+update(aUser : User) : User
+delete(aUser : User) : void
UserDAOImpl
UserPersistence
+read(aUserId : long) : User
+read(aUsername : String) : User
+update(aUser : User) : User
+delete(aUser : User) : void
UserPersistanceImpl

Figure 2: desing class diagramm UserServiceREST

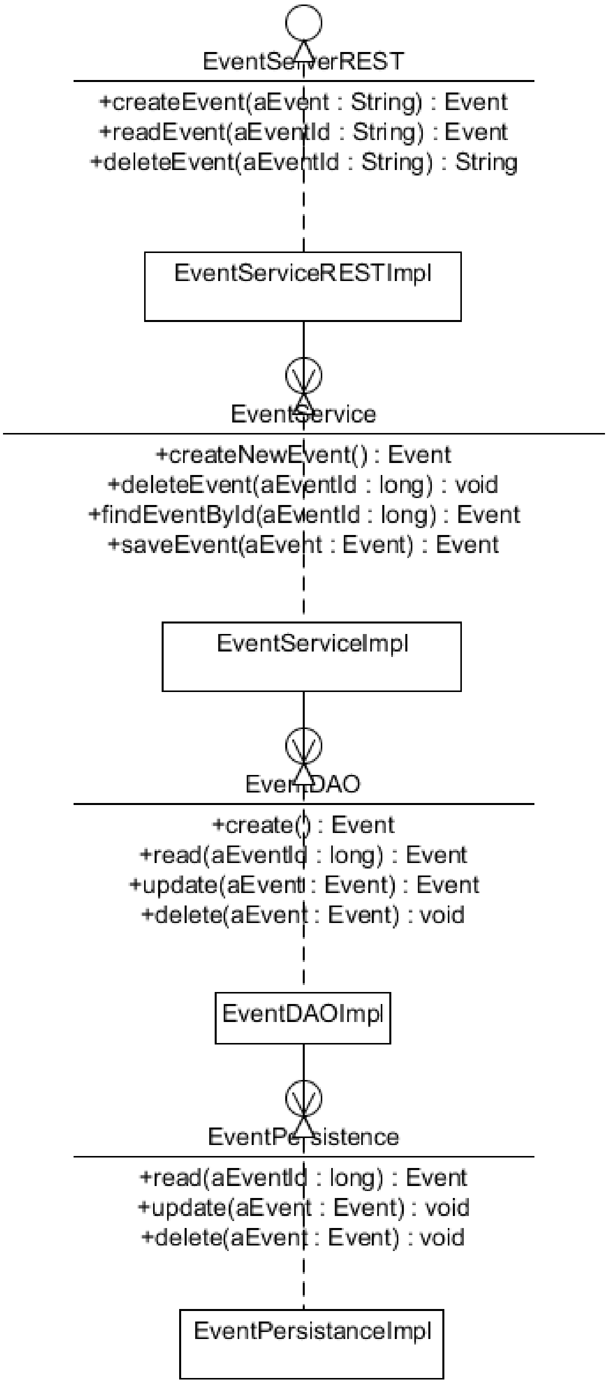
EventServerREST
+createEvent(aEvent : String) : Event
+readEvent(aEventId : String) : Event
+deleteEvent(aEventId : String) : String
EventServiceRESTImpl
EventService
+createNewEvent() : Event
+deleteEvent(aEventId : long) : void
+findEventById(aEventId : long) : Event
+saveEvent(aEvent : Event) : Event
EventServiceImpl
EventDAO
+create() : Event
+read(aEventId : long) : Event
+update(aEvent : Event) : Event
+delete(aEvent : Event) : void
EventDAOImpl
EventPersistence
+read(aEventId : long) : Event
+update(aEvent : Event) : void
+delete(aEvent : Event) : void
EventPersistanceImpl

Figure 3: desing class diagramm EventServiceREST

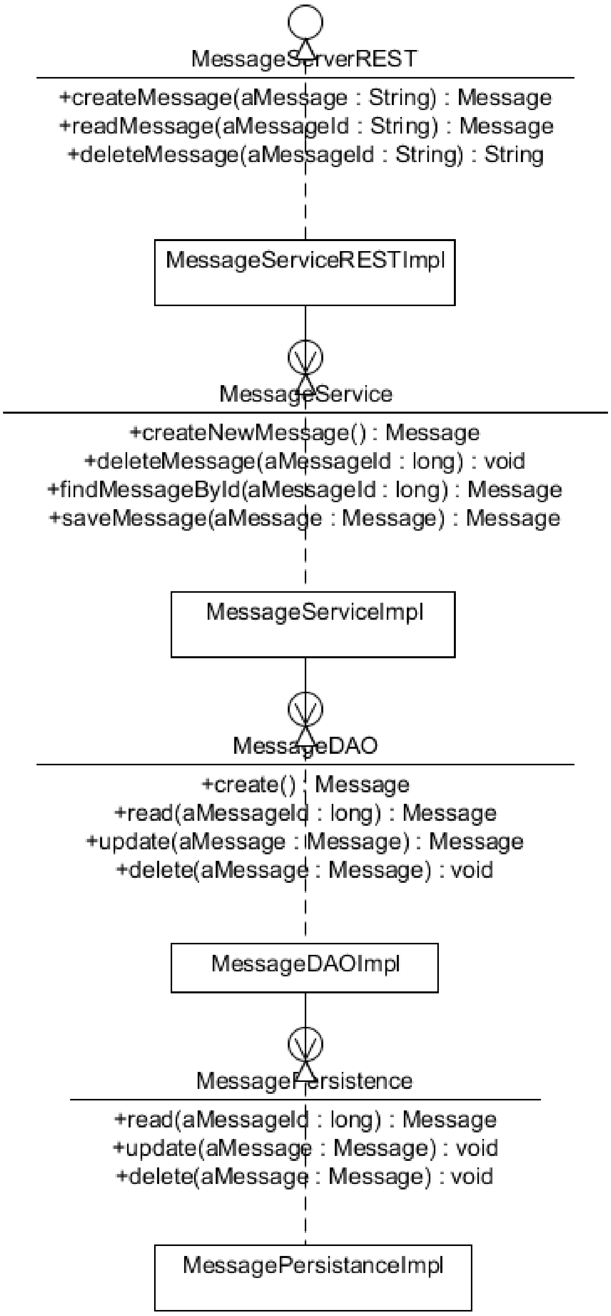
MessageServerREST
+createMessage(aMessage : String) : Message
+readMessage(aMessageId : String) : Message
+deleteMessage(aMessageId : String) : String
MessageServiceRESTImpl
MessageService
+createNewMessage() : Message
+deleteMessage(aMessageId : long) : void
+findMessageById(aMessageId : long) : Message
+saveMessage(aMessage : Message) : Message
MessageServiceImpl
MessageDAO
+create() : Message
+read(aMessageId : long) : Message
+update(aMessage : Message) : Message
+delete(aMessage : Message) : void
MessageDAOImpl
MessagePersistence
+read(aMessageId : long) : Message
+update(aMessage : Message) : void
+delete(aMessage : Message) : void
MessagePersistanceImpl

Figure 4: desing class diagramm MessageServiceREST

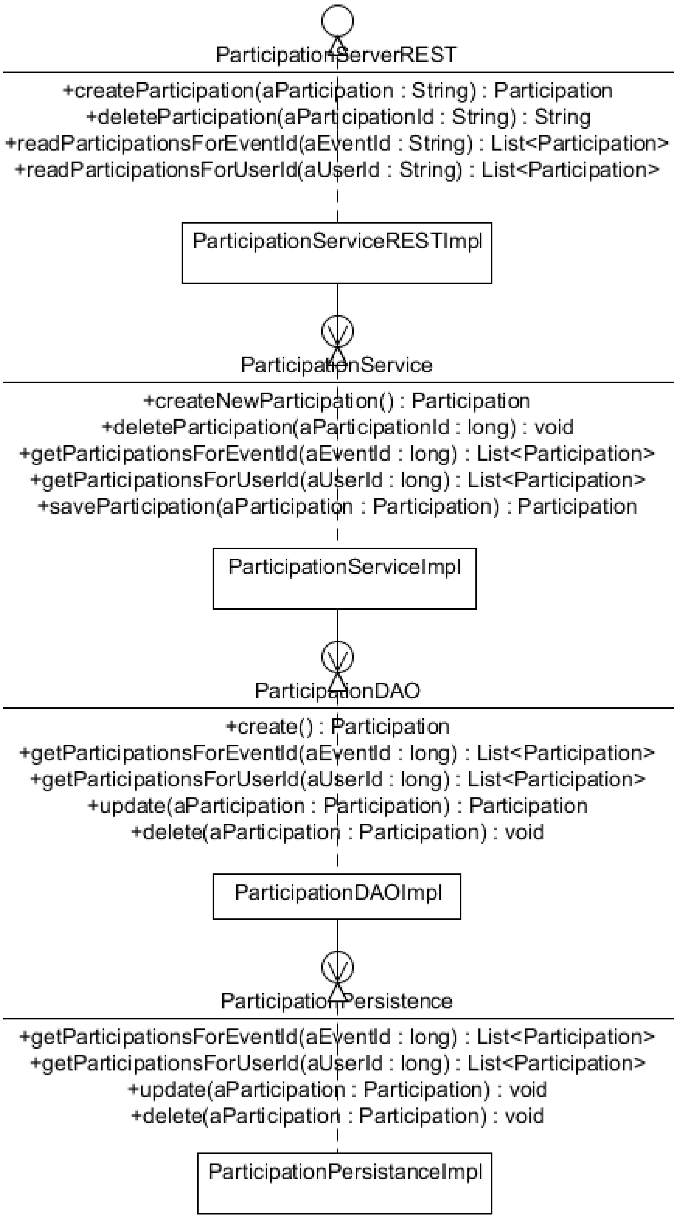
ParticipationServerREST
+createParticipation(aParticipation : String) : Participation
+deleteParticipation(aParticipationId : String) : String
+readParticipationsForEventId(aEventId : String) : List<Participation>
+readParticipationsForUserId(aUserId : String) : List<Participation>
ParticipationServiceRESTImpl
ParticipationService
+createNewParticipation() : Participation
+deleteParticipation(aParticipationId : long) : void
+getParticipationsForEventId(aEventId : long) : List<Participation>
+getParticipationsForUserId(aUserId : long) : List<Participation>
+saveParticipation(aParticipation : Participation) : Participation
ParticipationServiceImpl
ParticipationDAO
+create() : Participation
+getParticipationsForEventId(aEventId : long) : List<Participation>
+getParticipationsForUserId(aUserId : long) : List<Participation>
+update(aParticipation : Participation) : Participation
+delete(aParticipation : Participation) : void
ParticipationDAOImpl
ParticipationPersistence
+getParticipationsForEventId(aEventId : long) : List<Participation>
+getParticipationsForUserId(aUserId : long) : List<Participation>
+update(aParticipation : Participation) : void
+delete(aParticipation : Participation) : void
ParticipationPersistanceImpl

Figure 5: desing class diagramm ParticipationServiceREST

6. DATABASE

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

7. PRELIMINARIES

8. PREVIOUS WORK

9. SORTING

...

10. SHORTEST PATHS

...

11. EULERIAN CYCLES

...

11.1 Edge Walks

...

11.2 Restricted Mutation Operators

...

11.3 Adjacency List Matchings

...

12. CONCLUSIONS

Acknowledgments

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

13. REFERENCES

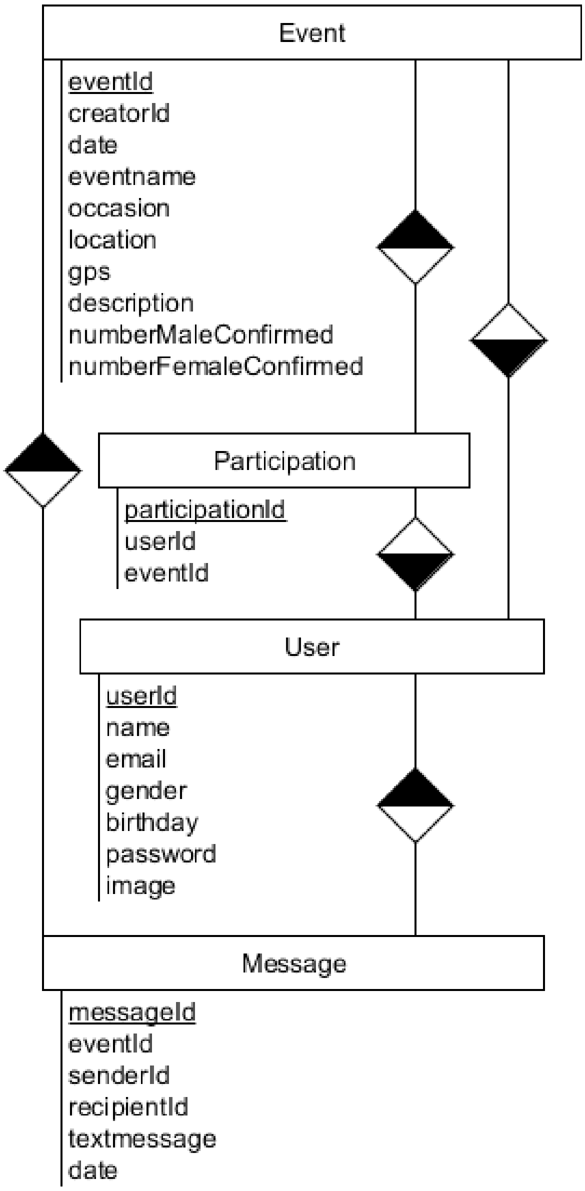

Figure 6: eer-diagram