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.

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. 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.

6. 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
- JUnit 4.8.2
- Maven 3
- Google Guice 3
- UMLet 11.3

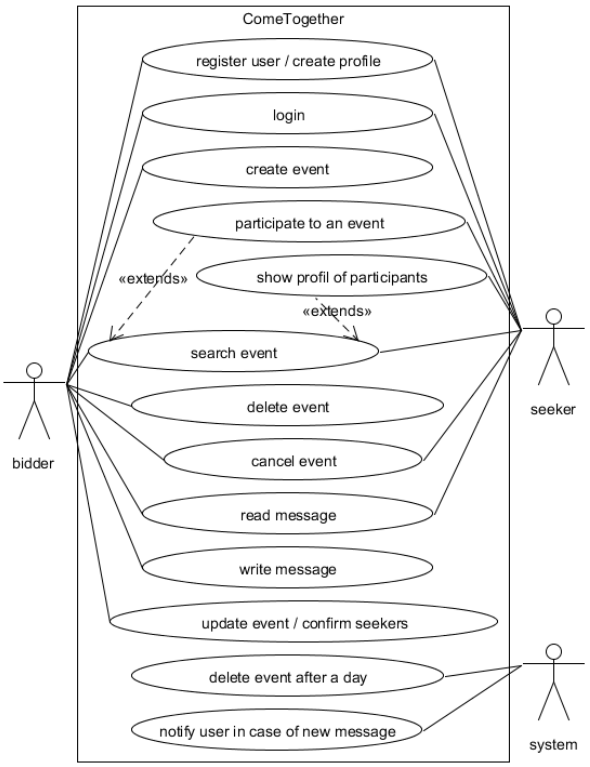

Figure 1: use case diagram

- EasyMock 3.0
- JBoss Resteasy 2.2.1
- Jackson 1.9.2
- apache http 3.1
- PostgreSQL 9.1.4

7. 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.

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.

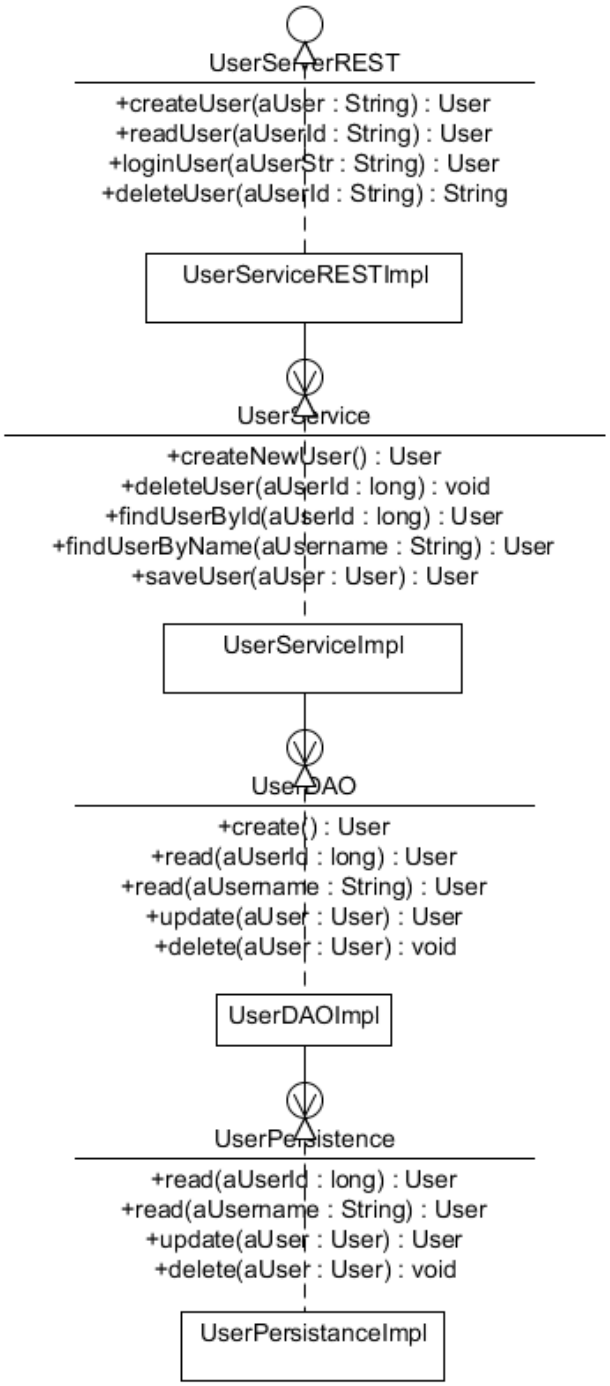

Figure 2: desing class diagramm UserServiceREST

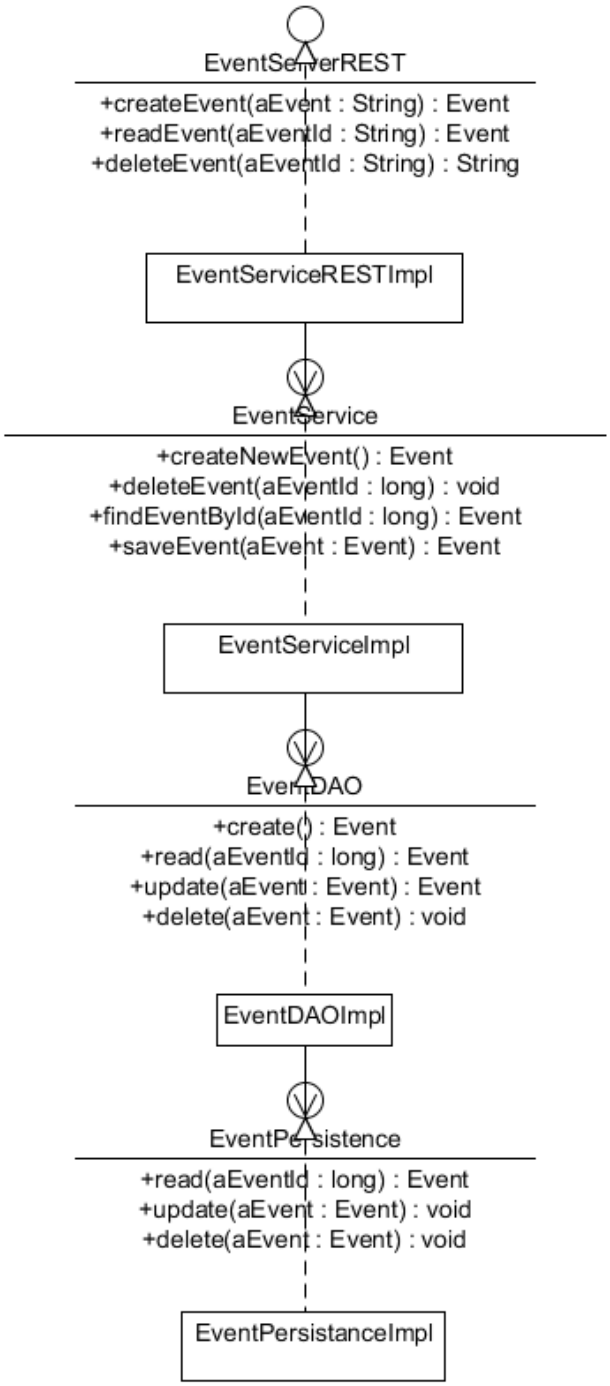

Figure 3: desing class diagramm EventServiceREST

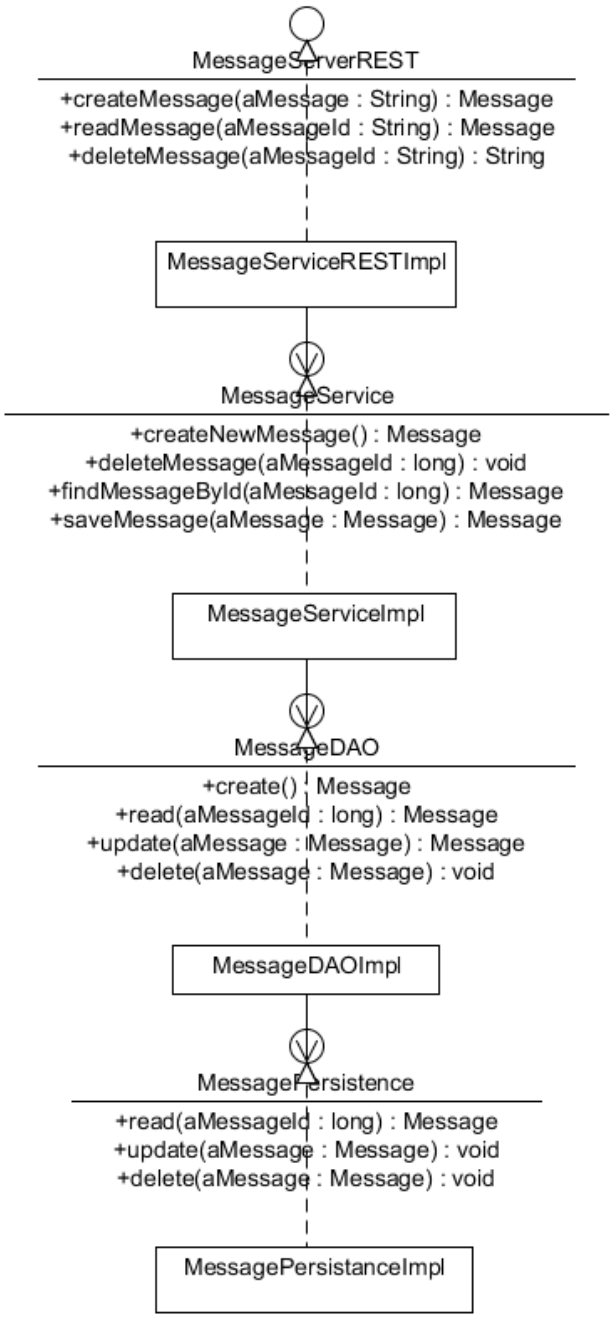

Figure 4: desing class diagramm MessageServiceREST

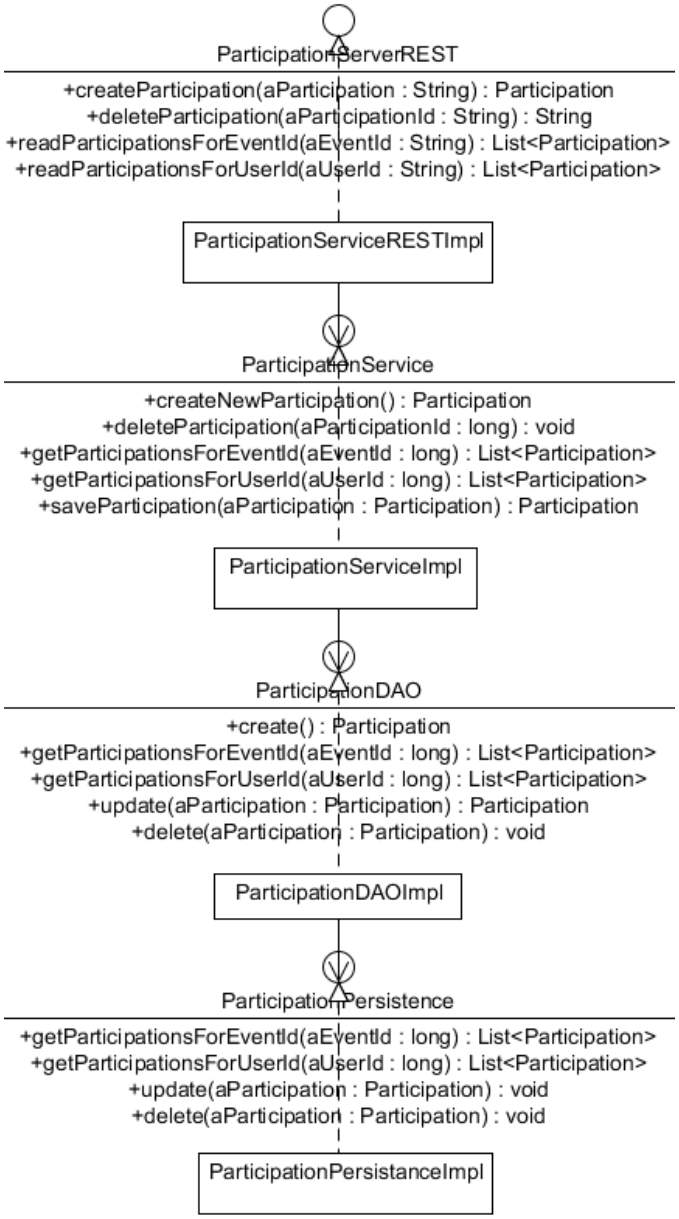

Figure 5: desing class diagramm ParticipationServiceREST

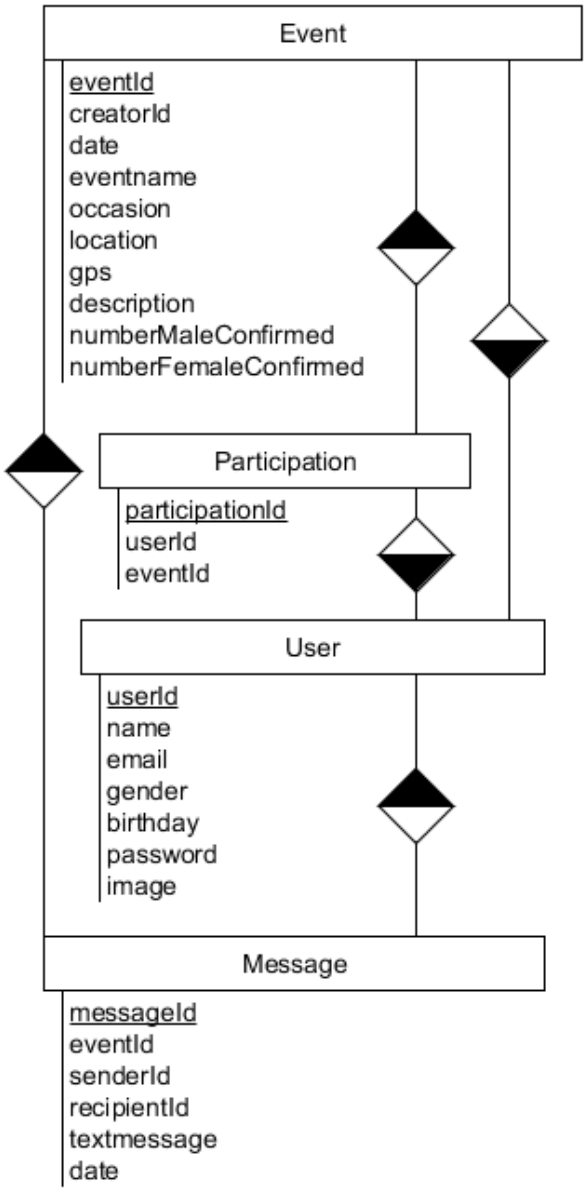

Figure 6: eer-diagram

8. 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 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.

9. FRONTEND

9.1 User Interface Design

Eine Webseite, wie auch eine App kommt bei den Nutzern gut an, wenn sie benutzerfreundlich und bedienbar ist. Auch wenn die Anwendung die besten und neusten Funktionen bietet, wird der User daran wenig Spass haben, wenn er die Features nicht schnell findet oder lange navigieren muss um zu seinen gewünschten Inhalten zu gelangen. Schon in den 90er Jahren hat sich Jakob Nielsen, ein dänischer Schriftsteller und mittlerweile eine der führenden Persönlichkeiten auf dem Gebiet der Benutzerfreundlichkeit, ausführlich mit dem Themen Webdesign und Usability beschäftigt. In diesem Zusammenhang erstellte er eine Liste mit zehn Grundprinzipien, die man bei der Gestaltung von Bedienoberflächen beachten sollte. Die Grundprinzipien wurden mit dem Gedanken an die Gestaltung von Webseiten erstellt, gelten aber zum Teil genauso für Apps (http://www.useit.com/papers/heuristic/heuristic_list.html).

9.2 Design einer Android-App

Auf der Google I/O 2010 präsentierte Jim Palmer das erste Mal Richtlinien, an die man sich halten kann bei der Entwicklung von Android-Apps. Sie beschreiben wie eine App auszusehen hat:

- einfach und verständlich
- sich auf den Inhalt konzentrieren und nicht nur auf das Aussehen
- konsistent sein, damit sich der Nutzer schnell zurechtfindet

Bei der Entwicklung der ComeTogether-App war es uns wichtig sich an diese Guidelines zu halten und somit mussten die uns zur Verfügung stehenden GUI-MockUps der zu entwickelnden App angepasst werden, bevor wir mit der Umsetzung des Frontends beginnen konnten. Dafür benutzen wir das Tool Balsamiq Mockups. Per Drag & Drop lassen sich ganz einfach GUI-Element(Formularfelder, Navigationsleisten, Tabs usw.) auf einer Arbeitsfläche zusammenklicken. Auf der Webseite mockupstogo gibt es für Android oder iOS Erweiterungen falls die Standardelemente nicht ausreichen.

Figure 7: Balsamiq Mockups

9.3 Sherlock ActionBar

Ab Android 3.0(Honeycomb) ersetzt die Action Bar das Options-Menü(als Popup-Menü mit der Menü-Taste aufrufbar) sowie die Title Bar. Die Action Bar erlaubt Menus und Buttons zu platzieren und bietet die Möglichkeit sich in einen oberen und einen unteren Teil aufzuteilen. Die Action Bar ist immer am Screen sichtbar(kann mit hide() unsichtbar gemacht werden) und macht somit das User Interface

konsistent, was viel dazu beiträgt das Design der App benutzerfreundlich zu gestalten.

Um die Action Bar auf Android 2.x nutzen zu können

Figure 8: Frontend

wurde die Open-Source-Library ActionBarSherlock entwickelt. Dabei handelt es sich um eine Erweiterung der Support Library, die es ermöglicht APIs auf älteren Android-Platformen zu nutzen.
(http://developer.android.com/tools/extras/support-library-.html)
Mit der neusten Version der ActionBarSherlock(v4.1.0) soll es dem App-Entwickler möglich sein, die Action Bar aus Android 4.0(Ice Cream Sandwich) mit allen Features auf 2.x nutzbar zu machen. Auf der Projektseite von ActionBarSherlock gibt es einige Demos, sowie einen Link zur GitHub-Seite mit allen Quellcodes. Um ein Android-Projekt mit der ActionBarSherlock aufsetzen zu können geht man wie folgt vor:

1. Nachdem man ActionBarSherlock heruntergeladen hat, findet man einen Ordner "'library"', den man als "'new Android project - from existing source"' in Eclipse importieren muss. In diesem Projekt klickt man unter Properties/Android das Häkchen bei "'Is library"' an.

2. Im eigentlichen Android-Projekt geht man in Properties/Android und klickt auf add. Daraufhin sollte die ABS-library erscheinen und in das Projekt eingebunden werden können.

3. In der AndroidManifest.xml muss für jede activity, die die ABS haben soll, der style festgelegt werden mit der folgenden Zeile:
android:theme="'@style/Theme.Sherlock.Light"'>

4. In dem activities wird die ABS mit extends vererbt und mit der folgenden Methode aufgerufen:

```
onCreateOptionsMenu(Menu menu) {}
```

9.4 Hintergrundoperationen

9.4.1 REST-Kommunikation

Um auf die Methoden des RESTful-Webservice zugreifen zu können werden zuerst die Instanzen von dem jeweiligen Datentyp wie beispielsweise User mit allen notwendigen Daten befüllt, anschliessend wird eine Instanz des User-Client erstellt und deren Methode createUser(user) aufgerufen. Die Methode createUser erbt von der abstrakten Klasse Creator und initialisiert eine Verbindung zum Webservice mit initConnection("'.../ctUser/createUser"') um anschliessend die Methode writeProduct aufzurufen, die schliesslich post.setEntity(new StringEntity(product)) ausführt und die Daten an den Server per POST sendet.

9.4.2 UI-Thread und langlaufende Programmteile

UI-Thread ist in einer Android-App für die Darstellung und Anwendereingaben zuständig. Alle Methoden einer Android-Komponente, die mit "'on"' beginnen, wie z.B. onClick laufen in diesem UI-Thread und dürfen nicht blockieren, das bedeutet, dass alle Methoden, die auf Oberflächenereignisse reagieren, schnell abgearbeitet werden müssen(innerhalb fünf Min.). Werden sie nicht innerhalb dieser Zeitspanne abgearbeitet, und der Nutzer drückt z.B. auf die Menütaste, dann geht Android davon aus, dass die Anwendung abgestürzt ist und der Nutzer bekommt einen Application not responding(ANR) angezeigt. Eine Lösung hierfür wäre ein eigener Thread, der in dieser onClick-Methode ausgeführt wird, damit die Anwendung möglichst schnell auf Nutzereingaben reagiert und nicht blockiert. Das Problem dabei ist, dass die Anzeige nur der UI-Thread verändern darf und jeder Zugriff von ausserhalb zu einer Exception führt. Man bräuchte noch eine zusätzlichen Thread, der mit Hilfe eines Handlers eine Nachricht dem UI-Thread übergibt und dieser die Informationen aus dem ersten Thread holt und anzeigt. Denn es ist anderen Threads nicht erlaubt, Daten im UI-Thread zu verändern, da eine Veränderung des UI-Threads die Synchronisierung der Objektzustände erforderlich machen würde und somit nicht automatisch ausgeführt wird. Das ganze ist mit Threads sehr kompliziert und wird in Android auf eine einfachere Art und Weise mit Hilfe von AsyncTask gelöst.

9.5 AsyncTask

AsyncTask ist eine Klasse, die es ermöglicht Operationen im Hintergrund zu bearbeiten und anschliessend einen Code im UI-Thread auszuführen. Um die Klasse zu nutzen muss man eine Unterklasse von AsyncTask erstellen. AsyncTask enthält drei Methoden, die in der Unterklasse überschrieben werden müssen[2]:

- doInBackground():
  Hier steht der Code, der im Hintergrund, in einem separaten Thread, ausgeführt werden soll. Beim Aufruf mit execute() kann der Methode als Parameter ein String übergeben werden, der dann in der eigentlichen Methode in ein String-Array umgewandelt wird.

- onProgressUpdate():
  Dieser Code-Abschnitt wird ausgeführt, wenn es einen Fortschritt bei der Ausführung gibt, der aus der doInBackground-Methode gemeldet wird. Der Fortschritt kann beispielsweise der prozentuale Anteil einer heruntergeladenen Datei sein.

- onPostExecute():
  Diese Methode wird aufgerufen, wenn die doInBackground-

Methode abgearbeitet ist. Es handelt sich um die Benutzerschnittstelle, die dem Nutzer meldet, dass die Aufgabe erledigt worden ist. Das Argument dieses Methodenaufrufs ist der Parameter, den die doInBackground-Methode zurückliefert.

Um den AsyncTask auszuführen wir die Unterklasse instantiiert und die Methode execute() aufgerufen. Die Frage ist nun wann man AsyncTask einsetzen sollte. Ab besten ist es so viel wie möglich in den Hintergrund zu verschieben, denn der UI-Thread führt nur Änderungen an der Anzeige aus, wenn er nicht beschäftigt ist. Deswegen sollten Aufrufe, die etwas längern dauern, wie beispielsweise der Datenbankzugriff, ein Webservice-Aufruf, das Parsen eines JSON-Objekts usw. in einem AsyncTask aufgerufen werden. Mit einem Ladebalken kann dem Nutzer mitgeteilt werden wie der Fortschritt der Aufgabe ist und ihm dadurch das Gefühl vermitteln, dass die App schnell und flüssig läuft. Mit der Methode isFinishing() kann in onPostExecute() geprüft werden, ob die Activity, in der der AsyncTask läuft nicht mittlerweile beendet worden ist, weil es kann vorkommen, dass der Nutzer die Activity geschlossen hat, aber der AsyncTask noch läuft und versuchen wird die onPostExecute() auszuführen.
In der ComeTogether-App nutzen wir die Vorteile der AsyncTask u.a. bei Zugriffen auf den Webservice.

9.6 Übergabe von Objekten zwischen Activities

Nachdem ein User auf ComeTogether sich erfolgreich eingeloggt hat, liefert der Webservice ein Objekt der Klasse User. Dieses Objekt muss zwischen Activities übertragen werden und jederzeit zur Verfügung stehen. Es gibt mehrere Möglichkeiten dieses Problem zu lösen[1]:

1. Interface Parcelable implementieren:
   Damit die Instanzen der Klasse User zwischen Activities ausgetauscht werden können, muss die Klasse User das Interface Parcelable implementieren. Anschliessend kann man ganz einfach mit einem Intent die Objekte an die andere Activity weiterreichen, z.B. intent.putExtra("‘user"’, new User());
   Die Daten bokommt man aus dem Bundle folgendermassen:
   Bundle data = getIntent().getExtras();
   User user = data.getParcelable("‘user"’);

2. Objekt serializieren:
   Eine weitere, aber weniger ressourcenschonende Möglichkeit bietet java.io.Serializable. Die Klasse User müsste das Interface Serializable implementieren. Anschliessend braucht man einen neuen Intent und Bundle. Mit der putSerializable("‘user"’, new User()) wird das Objekt dem Bundle hinzugefügt und mit intent.putExtras(bundle) das Bundle an das Intent angehängt. Mit startActivity(intent) wird die nächste Activity gestartet und das Objekt kann mit
   User user = (User)getIntent().getSerializableExtra("‘user"’)
   geholt werden.

3. In der SQLite Datenbank speichern:
   Als beste und einfachste Möglichkeit das User-Objekt allen notwendigen Activities jederzeit verfügbar zu machen erwiess sich die Speicherung der User-Daten intern in der SQLite Datenbank. Man kann sie aus jeder Activity heraus abfragen, die Daten bearbeiten und wieder an den Webservice übertragen, der sie in der PostgreSQL-Datenbank aktualisiert.

10. MAVEN-ANDROID-PROJEKT

Es war für uns selbstverständlich bei der Entwicklung vom Backend Maven zu nutzen, doch von der Entwicklung eines Android-Projekts auf Basis von Maven hat man uns abgeraten, da es zu vielen Problemen kommen kann. Auch aus zeitlichen Gründen haben wir uns mit Maven im Zusammenhang mit Android nicht mehr viel beschäftigt. Dennoch möchten wir hier erwähnen, dass es auf http://code.google.com/p/maven-android-plugin/ ein android-maven-plugin gibt, welches ein APK-Archiv erzeugt. Auf der gleichen Seite findet man auch dazu eine Dokumentation und Links zu Samples auf gitHub.

11. 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.

It was no time to read conference publication or journal publications. Solutions were mostly found on normal blogs.

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.
Da wir bisher im Bachelor-Studium nur ganz wenig mit Android zu tun hatten, mussten wir uns in die App-Programmierung einarbeiten und an kleinen Beispielprojekten die verschiedenen Android-Mechanismen wie z.B. den AsyncTask, ActionBar und die SQLite-Datenbank kennenlernen bevor wir richtig anfangen konnten die ComeTogether-App zu entwickeln. Es gab auch in der Frontend-Entwicklung immer wieder Probleme, die lösbar waren aber viel Zeit gekostet haben und wir nicht richtig vorangekommen sind. Wir haben jetzt die wichtigsten Grundbausteine für die Entwicklung der ComeTogether-App geschaffen, dazu gehört u.a. die Kommunikation mit dem RESTful-Webservice, der Zugriff auf die SQLite-Datenbank als auch die Einsatzmöglichkeiten von AsyncTasks. Der nächste Schritt wäre, diese Grundbausteine zu nutzen und Schritt für Schritt die restlichen Activities und Views zu erzeugen, die uns bisher fehlen.

12. REFERENCES

[1] A. Becker. *Android 2, Grundlagen und Programmierung.* Dpunkt Verlag, 2010.

[2] M. Gargenta. *Einfuehrung in die Android-Entwicklung.* O'Reilly, 2011.