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 no flirt or 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. REQUIREMENTS ENGINEERING

Figure 1 shows the use case diagram for ComeTogether.

3. 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.

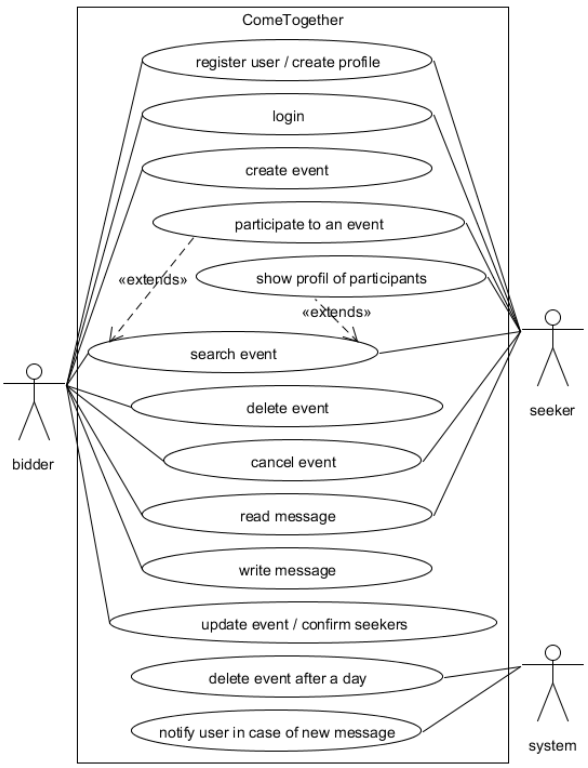

Figure 1: use case diagram

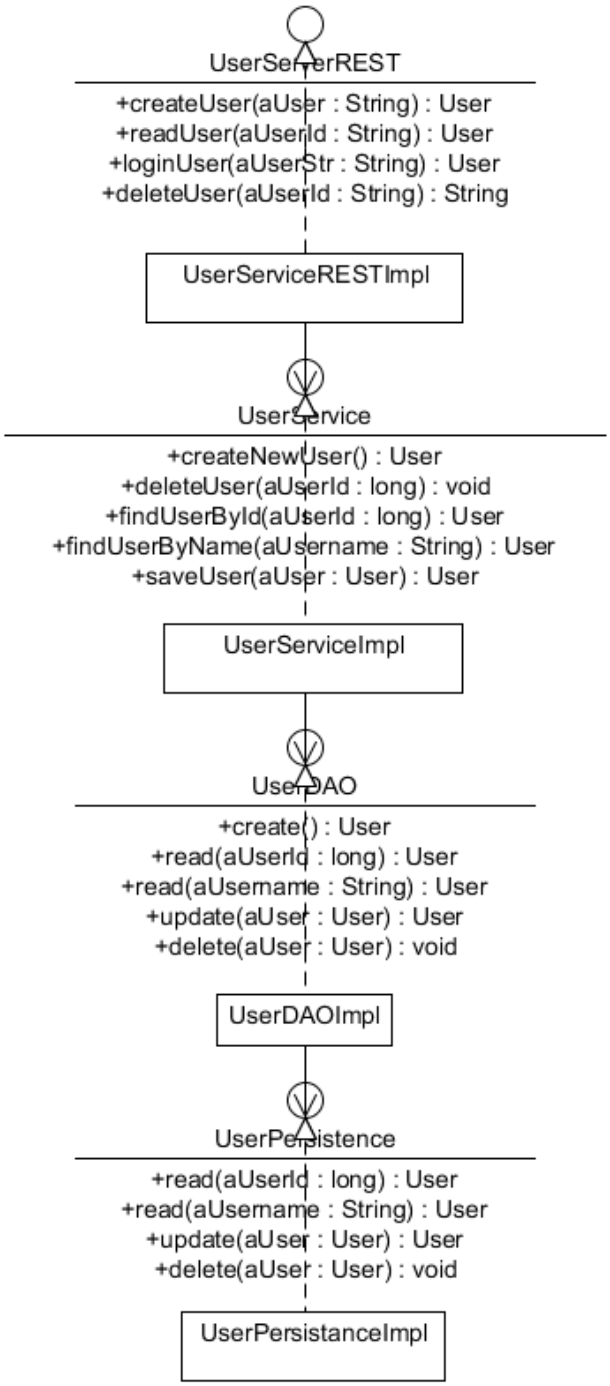

Figure 2: desing class diagramm UserServiceREST

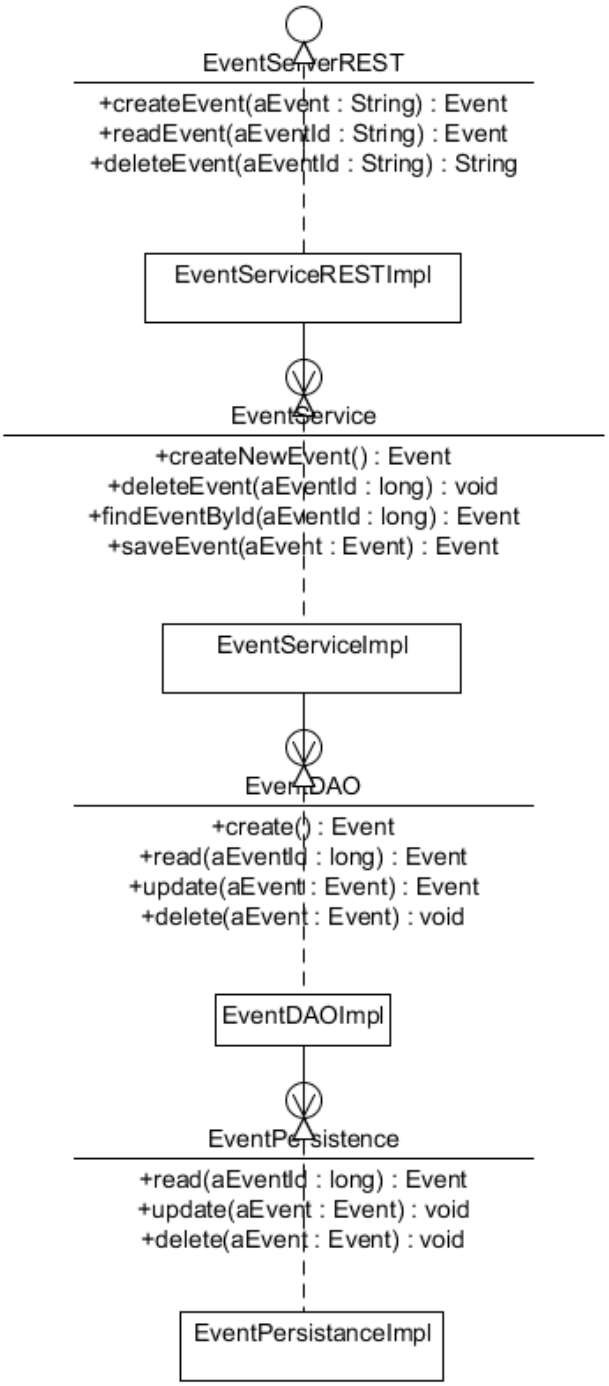

Figure 3: desing class diagramm EventServiceREST

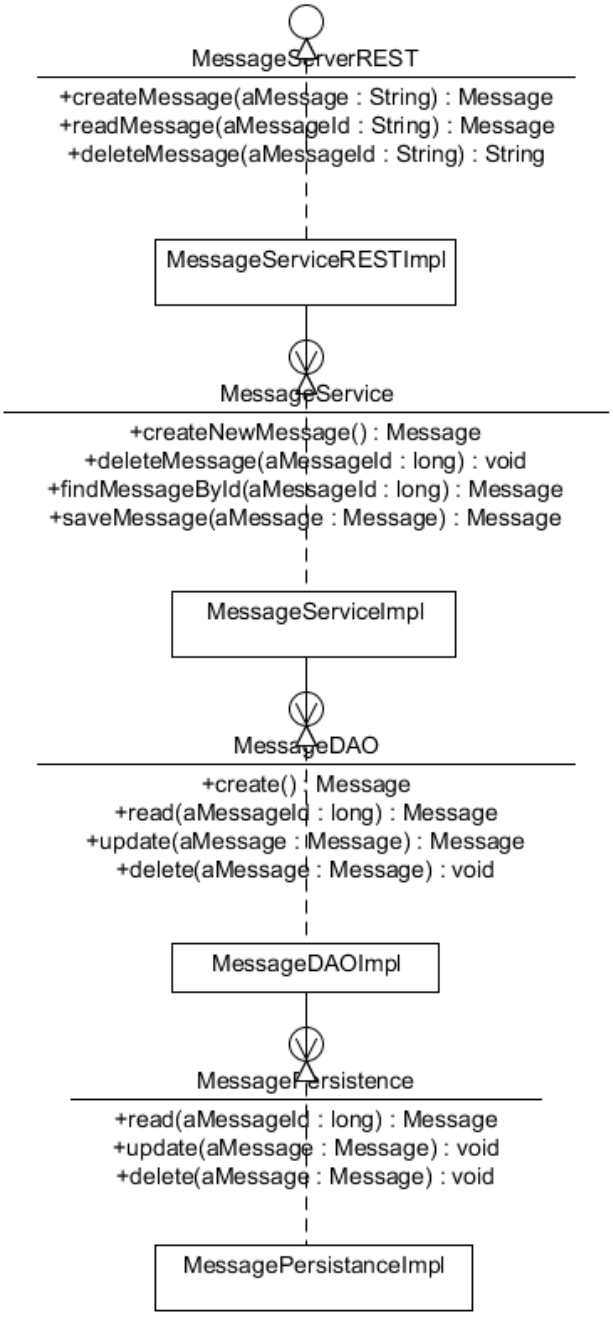

Figure 4: desing class diagramm MessageServiceREST

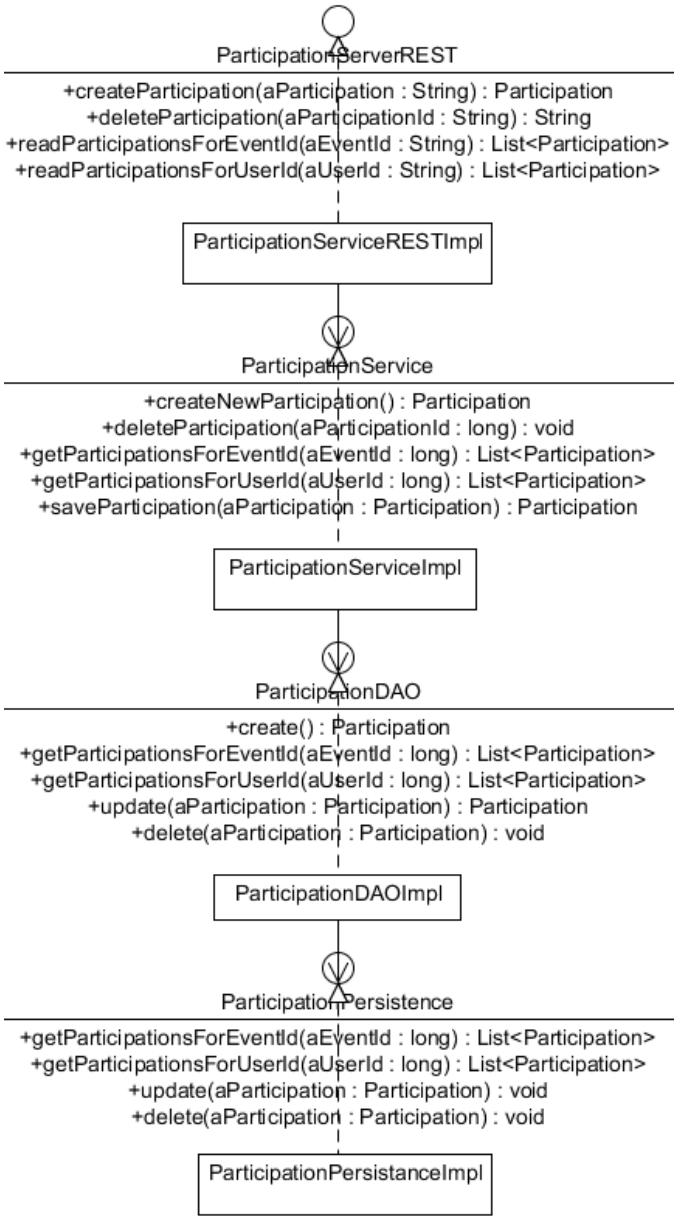

Figure 5: desing class diagramm ParticipationServiceREST

4. DATABASE

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

5. 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.

5.1 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 zurechtfinden kann

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.

5.2 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 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. 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

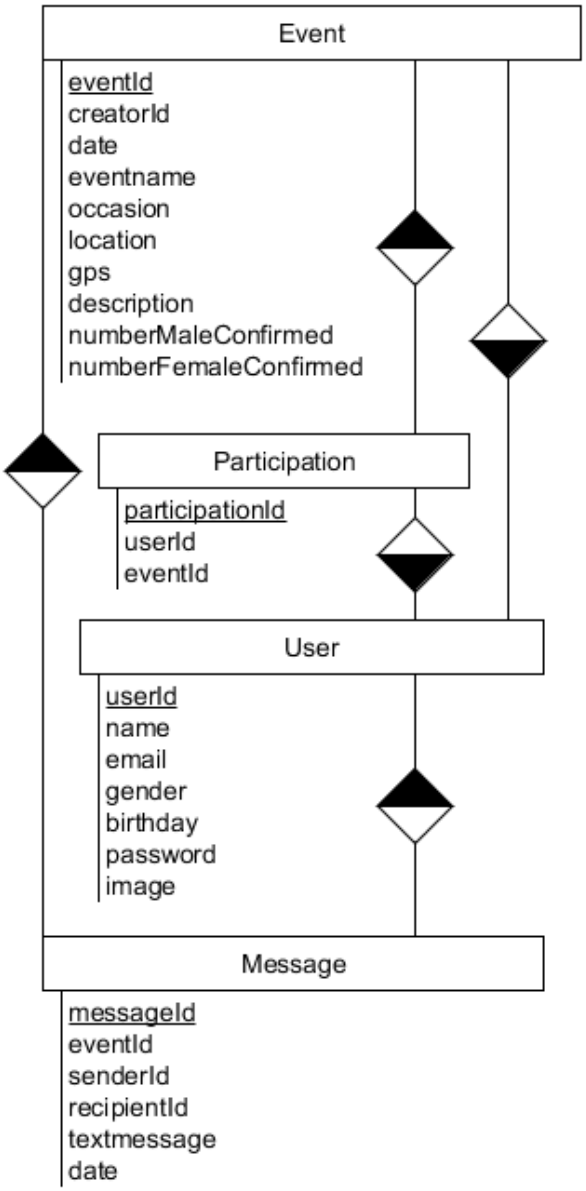

Figure 6: eer-diagram

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:

```
public boolean onCreateOptionsMenu(Menu menu) {
   ...
   }
```

5.3 Hintergrundoperationen

5.3.1 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 UserClient erstellt und deren Methode createUser(user) aufgerufen. Die Methode createUser erbt von der abstrakten Klasse Creator und initialisiert eine Verbindung zum Webservice mit initConnection("'http://10.0.2.2:8080/ctBackend/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.

5.3.2 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.

5.4 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:

- 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.

5.5 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.

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.

6. PRELIMINARIES

7. PREVIOUS WORK

8. SORTING

...

9. SHORTEST PATHS

...

10. EULERIAN CYCLES

...

10.1 Edge Walks

...

10.2 Restricted Mutation Operators

...

10.3 Adjacency List Matchings

...

11. CONCLUSIONS

Acknowledgments

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

12. REFERENCES