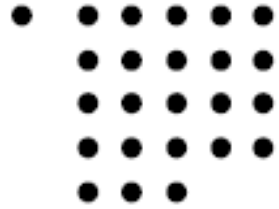


Fachhochschule Köln
Cologne University of Applied Sciences
Fakultät für Informatik und Ingenieurwissenschaft

Studiengang Medieninformatik
Lehrveranstaltung: Entwicklungsprojekt interaktive Systeme im SS 2015

Meilenstein 4 “ Autism “

vorgelegt von:

Jan Freundlieb

Irene Janzen

Betreuer:

Prof. Dr. Kristian Fischer

Prof. Dr. Gerhard Hartmann

B. Sc. Robert Gabriel

Köln, April 2015

Inhaltsverzeichnis

1 Datenstruktur

In diesem Kapitel wird auf der Grundlage der Anforderungsspezifikation, siehe (GITHUB LINK), die Datenstruktur definiert, die eine Basis für Algorithmen darstellt. Des Weiteren folgt ein Datenmodell, siehe Abbildung 1, dass die Beziehung zwischen der einzelnen Entitäten zeigt um zu verdeutlich in welchen Zusammenhang diese stehen.

Kategorie	**BESTEHT AUS**
name:	String
unterKategorie:	Liste Mit Kategorie
Situation:	Liste Mit Situation

Situation:	**BESTEHT AUS**
mitarbeiterImKontext:	User
ort:	String
situationsBeschreibung:	String
handlungsOptionen:	Liste Mit Handlungsoptionen
Kategorie	Liste Mit Situation

Handlungsoption:	**BESTEHT AUS**
schritte:	String
situation:	Liste Mit Situationen
rueckmeldungFuerCoach:	String
handlungsprofil:	Handlungsprofil
begruendung:	String
Ergaenzung	String

Handlungsprofil	**BESTEHT AUS**
handlungsOptionen:	Liste Mit Handlungsoptionen
autist:	User

User	**BESTEHT AUS**
email:	String
vorname:	String
nachname:	String
betrieb:	String
alter:	String
rolle:	Userrolle

Userrolle	**BESTEHT AUS**
name:	String
User	Liste Mit User

Bewertung	**BESTEHT AUS**
handlungsoption:	Handlungsoption
situation:	Situation
Mitarbeiter	User

autist:	User
sozialerKontextBewertung:	Liste Mit Strings
bemerkung:	String

Termin	**BESTEHT AUS**
betreff:	String
startDatum:	Date
endDatum	Date
bemerkung:	String

Abrufstatistik	**BESTEHT AUS**
kategorie:	Kategorie
zeit:	Date

sozialerKontextStatistik	**BESTEHT AUS**
bewertung:	Bewertung

1.1 Datenmodell

Auf der Basis der detailreichen Datenstruktur und des Datenmodells werden die XML-Schemata erstellt.

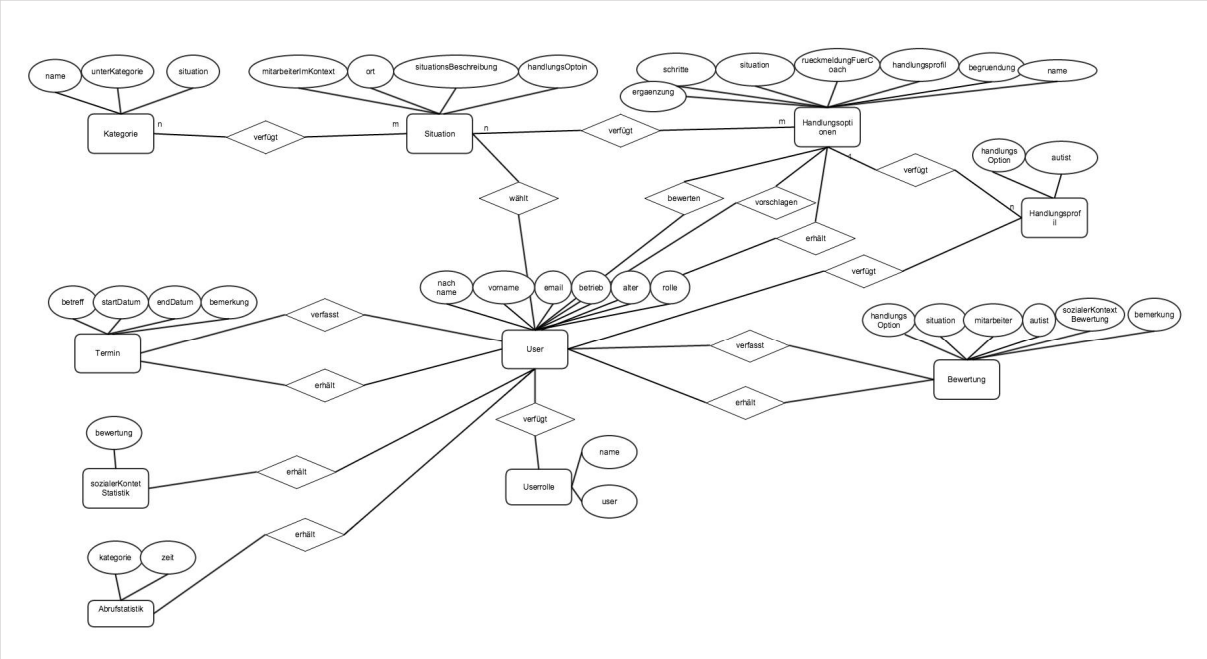


Abbildung 1: Datenmodell

1.2 Queues

Queues werden innerhalb von RabbitMQ verwendet, um eintreffende Nachrichten zu verteilen. Es wird zwischen einem Producer und einem Consumer unterschieden, wobei ein Producer eine Nachricht an die Queue liefert und der Consumer diese Nachrichten ausliest. Im Folgenden sollen die Queues aufgeführt werden die für das zu entwickelnde System relevant sind. RabbitMQ bietet auch Worker Queues, wodurch das RPC Pattern unterstützt wird. Diese Queues werden hauptsächlich dazu verwendet den Datenbestand zu erweitern.

Notfallsituation:

Wenn es zu Situationen noch keine Handlungen gibt und der Autist Hilfe benötigt, kann er über die Queue „Notfallsituation“ mit dem Job Coach kommunizieren.

Überschneidung:

Bei einer Terminüberschneidung, kann der Autist über die Queue „Überschneidung“ andere Teilnehmer über sein Verlassen des Termins benachrichtigen.

Handlungsvorschlag (Worker Queue):

Über diese Queue hat der Job Coach die Möglichkeit Handlungsvorschläge

zu bestimmten Situationen anzulegen.

Bewertung:

Der Mitarbeiter kann Bewertungen an die Queue liefern, ein Consumer soll die Verarbeitung der eingehenden Bewertung übernehmen.

Kritisch:

Ist ein kritischer Wert erreicht so soll der Consumer über die Queue „kritisch“ den Job Coach informieren.

1.3 Algorithmen

Notfallsituation:

Eine Notfallsituation tritt ein wenn der Autist auf seiner Arbeit in Situationen gerät, in denen er nicht weiß wie er zu handeln hat. Hierfür wird über die Queue „Notfallsituation“ mit dem Job Coach kommuniziert. Im Folgenden werden die wichtigsten Codeabschnitte für den Consumer und den Producer aufgeführt.

Producer

```
...
// Mittels der übergebenen Parameter wird eine Verbingung mit dem
Message Broker RabbitMQ hergestellt
ConnectionFactory factory = new ConnectionFactory();
factory.setHost(host);
```

```
factory.setUsername(username);
factory.setPassword(password);
factory.setPort(5672);
Connection connection = factory.newConnection();
Channel channel = connection.createChannel();
channel.exchangeDeclare(EXCHANGE_NAME, "fanout", false);
// Festlegen an welche Queue die Nachricht geliefert werden soll
channel.queueDeclare(QUEUE_NAME, false, false, false, null);
String tempstr = "";
for (int i = 0; i < Message.length; i++)
     tempstr += Message[i];
// mittels der Methode basicPublish wird die Nachricht verschickt
channel.basicPublish(EXCHANGE_NAME, QUEUE_NAME, null,tempstr.get-
Bytes());
channel.close();
connection.close();
…
```

Consumer:

```
…
// Mit Hilfe der Klasse QueueingConsumer
// werden die Nachrichten empfangen,
// die Methode nextDelivery nimmt die
// eingehenden Nachrichten entgegen und gibt diese zurück

QueueingConsumer.Delivery delivery;
try {
    delivery = consumeMessage.nextDelivery();
    message = delivery.getBody();
…
```

Eingehende Email untersuchen nach Meeting:

Das Postfach soll stündlich nach neuen Mails durchsucht werden. Die Betreffe der Mails werden dann mit bestimmten Schlagwörtern wie„Meeting“ verglichen. Werden neue Termine anhand der Mails identifiziert, so sollen diese in den Kalender eingetragen werden. Die Idee ist hierbei einen Scheduler in Form eines AlarmManagers zu initialisieren, der diesen Job durch Initialisierung eines Service ausführt.

```
...
AlarmManager alarms = (AlarmManager) this.getSystemSer-
vice(Context.ALARM_SERVICE);
Intent intent = new Intent(getApplicationContext(),EmailRecei-
ver.class);
final PendingIntent pIntent = PendingIntent.
getBroadcast(this,0, intent, 0);
alarms.setRepeating(AlarmManager.RTC_WAKEUP,System.currentTimeMil-
lis(), 3600000, pIntent);
...
```

Mit Hilfe der API JavaMail werden im folgenden Codeabschnitt erst einmal alle Emails ausgelesen.

```
Properties props = new Properties();
props.setProperty("mail.store.protocol", "imaps");
try {
    Session session = Session.getInstance(props, null);
    Store store = session.getStore();
    /*
     * TODO: Make Config File
     */
store.connect("imap.gmail.com", "janfreundlieb@gmail.com", "janjan-
jan1");
            Folder inbox = store.getFolder("INBOX");
            inbox.open(Folder.READ_ONLY);
            Message[] msg = inbox.getMessages();

for (int i = 0, n = msg.length; i < n; i++) {
            Message message = msg[i];
```

```
            System.out.println("--------------------------------");
            System.out.println("Email Numer " + (i + 1));
            System.out.println("Betreff: " + message.getSubject());
            System.out.println("Von: " + message.getFrom()[0]);
            System.out.println("Text: " +
message.getContent().toString());
}
```

Auf Kalendereinträge und Terminüberschneidungen reagieren:

In Android besteht die Möglichkeit mittels Observer auf bestimmte Aktionen zu reagieren. In dem Fall der Kalenderänderungen, handelt es sich um die Aktion CALENDAR_WRITE. Auf diese wird reagiert und der Kalendereintrag kann dann über einen Cursor, über den SQL-Abfragen abgesetzt werden können und ermittelt werden.

```
...
Cursor cur = getContentResolver().query(Uri.parse("con-
tent://com.android.calendar/events"), null, null, null, null);
cur.moveToLast();
...
```

Um herauszufinden, ob es durch den Kalendereintrag zu Überschneidungen mit anderen Terminen gekommen ist, kann mittels Cursor folgende SQL-Abfrage abgesetzt werden.

```
Cursor cur2 = getContentResolver().
query(Uri.parse("content://com.android.calendar/events"), null,
"_id != " + _id + " AND dtend >= " + dtstart + " AND dtstart <=" +
dtend, null,null);
```

Im späteren Verlauf soll mittels AlarmManager auf eine Überschneidung aufmerksam gemacht werden, sodass der Autist entsprechend reagieren kann. Befindet er sich zum Zeitpunkt der Überschneidung in einem Termin mit anderen Teilnehmern und er muss zu einem anderen Termin, so hat er die Möglichkeit an alle Teilnehmer, die denselben Termin haben, eine Nachricht über die Queue „Überschneidung“ zu senden. Die Ermittlung der Teilnehmer die denselben Termin haben, soll über die Erfassung der Kalendereinträge der einzelnen Teilnehmer erfolgen.

Handlungsoptionen anlegen:

Der Job Coach soll die Möglichkeit haben auf entsprechende Situationen Handlungen vorzugeben, welche der Autist zu jeder Zeit studieren kann. Dazu ist eine Worker Queue „Handlungsvorschlag“ vorgesehen, welche dem Job Coach eine Bestätigung schickt, wenn der Handlungsvorschlag abgespeichert worden ist.

```
// Pseudocode, Client
RpcClient rpcClient = new RpcClient();
rpcClient.überWorkerQueue(Handlungsvorschlag);
response = rpcClient.call(Handlung);

// Pseudocode, Server
parameter = entnehmeNachrichtaus(Handlungsvorschlag)
response = rpcFunction(parameter);
publish(response);
```

Bewertung des sozialen Kontextes:

Die Mitarbeiter haben die Möglichkeit den sozialen Kontext zu bewerten. Hierfür soll der Mitarbeiter, welcher in diesem Fall als Producer agiert, die Möglichkeit haben seine Bewertung an die Queue „Bewertung“ zu schicken, ein Consumer soll diese Bewertungen aus der Queue lesen und bei zu vielen kritischen Bewertungen den Job Coach über die Queue „kritisch“ eine entsprechende Nachricht über zukommen lassen. Ansonsten sollen die Bewertungen in einer Datenbank abgespeichert werden.

```
// Pseudocode
If ( bewertung >=3 ) {
   publishToJobCoach(„Kritischer Wert erreicht“);
} else {
   insertIntoDB(bewertung);
}
```

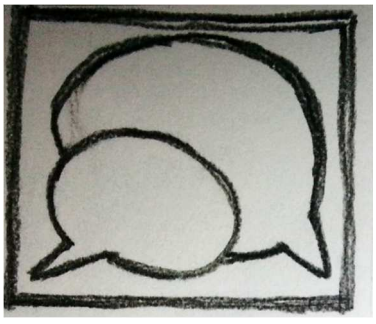
Abrufstatistiken und Bewertungsstatistiken ermitteln:

Damit der Job Coach durch Statistiken einen guten Überblick erhält, soll im Weiteren recherchiert werden, welche APIs Verwendung finden könnten.

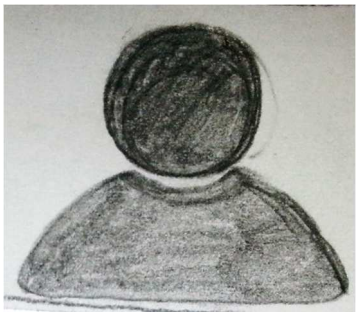
2 Metaphern und Paradigmen

Für alle Benutzer sollen die erstellten Metaphern ein grundlegendes Bestandteile und deren Beziehung verständlich sein, darum werden bereits bekannte Icons die sich aus der Marktrecherche ergaben und für das Team bekannte herangezogen, wie zum Beispiel Kalender. Jedoch enthält der Prototyp teilweise Symbole, die allgemein und universal von Benutzern anerkannt sind. Da das Verständnis der Icons auf frühere Erfahrung der Benutzer basiert und es eventuell Autisten (Novizen) gibt, die noch wenig Erfahrung haben, ist es sinnvoll diese im Vorfeld zu testen. Eine alternative Lösung wäre die Icons zusätzlich mit Beschriftung zu gestaltet, die die Bedeutung kommunizieren, um eine Mehrdeutig zu reduzieren oder gar auszuschließen. Aber gerade bei eine Mobilen Anwendung könnte es zu einer Reizüberflutung kommen, wenn zu viele Gestaltungselemente auf der Benutzeroberfläche vorhanden sind. Des Weiteren folgen einige Symbole, die vom dem Team gestaltet wurden, die vor Herausforderungen stellte, auf die nötigsten Formen zu reduzieren, die aber ausreichend waren, um die mit der Symbolik übertragenen Information zu übermitteln. Vor allem bei Autisten kommt es auf die eindeutige Symbolik an. Außerdem wird bei dem Icon-Design darauf geachtet, dass alle im selben Stil sind, die eine gemeinsame Perspektive und die gleiche Grundform besitzen, die somit auf eine Zusammengehörigkeit hinweisen.

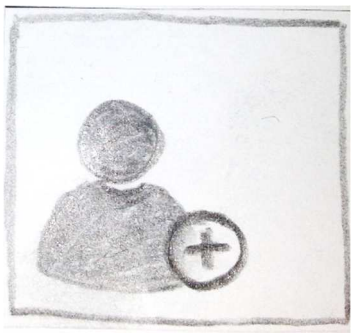
Interaktions-Stil ist eine Menüfläche die eine Auswahlmöglichkeit repräsentiert, die vorerst in grafischer Form vorliegt und eventuell später, je nachdem wie die Evaluierung ausfällt zusätzlich beschriftet wird. Anhand dieser Auswahl habe die Benutzer die Möglichkeit, die Aktivität vorzunehmen, die ihre Aufgabe erfüllen. Das Menü wird durch push buttons realisiert.



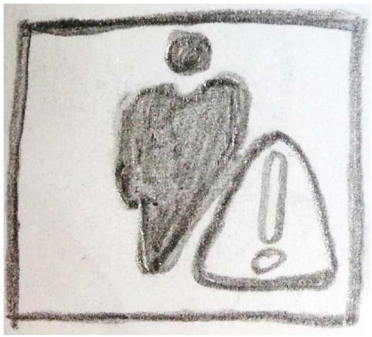
Metapher für Chat
Dies Aktivität soll dem Autisten die Möglcihkeit bieten dirket mit dem Job Coach zu kommunizieren.



Metapher für Handlungsprofil
Auf der Seite der Autisten bietet diese Aktivität die vorgeschlagenen Handlungen anzuschauen.
Auf der Seite des Job Coach gibt es eine Statistik darüber aus, für welche Bereiche die meisten Handlungsvorschläge nötig waren.



Metapher für Autisten Bewertung
Der Mitarbeiter kann den Autisten bewerten. Auf der Seite des Job Coach soll eine Auswertung anzeigen, welche Mitarbeiter, wie bewertet haben.



Metapher für Situationsbeschreibung
Der Autist kann unter dieser Aktivität seine Situation schildern mit der er Schwierigkeiten hat. Auf der Seite des Job Coach kann dieser für die geschilderten Situationen, Handlungsvorschläge verfassen.

Das Paradigma der Multimodalität -> textuell und Sprachaufzeichnungen
Interaktionsparadigma steht mit dem konzeptuellen Modell des Entwicklers zusammen
Interaktionsstil-> Formular->window oder panel->titel und thematisch strukturiert
Konversation-> Interaktions-Modus
Check-box

3 Abbildungsverzeichnis

4 Literaturverzeichnis