goApp Implementierung

Jörn Kussmaul, Katharina Riesterer, Julian Neubert,
Jonas Walter, Tobias Ohlsson, Eva-Maria Neumann

12. Februar 2017

Inhaltsverzeichnis

1 Einleitung

Anschluss auf Plichtenheft & Entwurf Allgemeine Beschreibung der App

2 Änderungen am Entwurf

Während der Impementierungsphase ergaben sich einige notwendige Änderungen gegenüber dem Entwurf. Diese sind im folgenden aufgeführt.

2.1 Server

2.1.1 Servlet

2.1.1.1 GoServlet Das GoServlet wird nicht mehr gebraucht und deswegen gelöscht. Die Funktionalität die diese Klasse haben sollte, befindet sich im nun Participate Servlet.

2.1.1.2 EventServlet Die Methode getEvent wurde durch getParticipates ersetzt. Im momentanen Zustand braucht unsere App keine eigene Methode getEvents über die eventId, da die Events über die Gruppe gelesen werden. Darüber bekommt man nur die Teilnehmer nicht, weswegen dafür eine eigene Methode benötigt wird.

2.1.1.3 GroupServlet Aus getGroups wurde getMembers, da die Gruppe schon beim Erstellen auf den Client geholt wird.

2.1.1.4 LocationServlet Die beiden Methoden setGPS, sowie setCluster werden zusammen über snycLocation aufgerufen.

2.1.1.5 Participate Servlet Die beiden Methoden accept und reject wurden durch die Methode setStatus ersetzt. Diese setzt den Status im Event-User Management, wodurch die Redundanz von zwei Methoden vermieden wird. Falls der Status Reject ist, wird der Teilnehmer gelöscht.

2.1.1.6 UserServlet Die Nutzer werden an den Client beim Einloggen bzw. Registrieren übermittelt und bis zum nächsten Einloggen gespeichert. Deswegen ist die Methode getUser überflüssig geworden.

2.1.1.7 ServletUtils Beim Implementieren der Servlets gibt es einige Schritte, die von jedem Servlet durchgeführt werden müssen. Dazu gehören zum Beispiel Methoden, um JSONObjekte aus der HTTP Anfrage auszulesen, oder das Verfassen der Antwort des Servers. Deswegen haben wir uns dazu entschieden eine statische Hilfsklasse zu nutzen, um diesen redundanten Code zu vermeiden. Außerdem definieren JSONObjekte eine wichtige Schnittstelle zum Client, da das Senden & Empfangen von Informationen durch diese geregelt wird. In unserem Fall bleiben Änderungen der Schnittstelle somit lokal.

2.1.1.8 JSONParameter Wir brauchen viel mehr Parameter um die JSONObjekte zu erstellen, als in der Entwurfsphase angenommen. Außerdem gibt es ein weiteres Enum für die vom Client aufrufbaren Methoden, sowie eines für Fehlercodes. Die JSONParameter sind auf dem Client, sowie Server gleich.

2.1.2 Algorithmus

2.1.2.1 Clusteralgorithmus Die Clusterfassade bietet nun neben der Methode getClusteredPoints auch die Methode getClusters und getCenter um direkt auf den Clusterer bzw. den Mittelpunktalgorithmus zugreifen zu können.

2.2 Model

2.2.0.1 Event Die Zeit eines Events wird nicht als Time sondern als Timestamp gespeichert, dann ist das Datum mit dabei.

2.2.0.2 Participant & Status Der Status in Participant ist zu einem Enum geworden. Der Status kann Inivted, Participate, Started oder Rejected sein.

2.3 Client

2.3.1 View

2.3.2 Controller

3 Kriterien

Im Pflichtenheft wurden Kriterien für unsere App definiert. Im folgenden unterteilen wir diese in erfüllt bzw. nicht erfüllt.

3.1 Erfüllte Kriterien

Alle Muss-Kriterien wurden erfüllt und außerdem noch einige von unseren Wunsch-Kriterien.

3.1.1 Kontoverwaltung

FA10 Anmeldung eines Benutzers in der App über Google Services

FA20 Ersterfassung und Änderung des Benutzernamens

3.1.2 Gruppenverwaltung

FA30 Jeder Benutzer kann eine Gruppe erstellen

FA35 Ersterfassung und spätere Änderung des Gruppennamens durch den Gruppengründer

FA40 Gruppen können über den Gruppennamen gesucht werden

FA45 Benutzer können Beitrittsanfragen an eine Gruppe senden

FA50 Der Gruppengründer kann Beitrittsanfragen der Gruppe verwalten:

- Anfragen bestätigen, wodurch der Anfragende ein Mitglied der Gruppe wird und die Anfrage gelöscht wird
- Anfragen ablehnen, wodurch die Anfrage gelöscht wird
- Anfragen ignorieren, wodurch die Anfrage bestehen und sichtbar bleibt

FA60 Der Gruppengründer kann Mitglieder aus der Gruppe entfernen

FA70 Die Gruppe kann durch den Gruppengründer gelöscht werden

FA80 Mitglieder der Gruppen, ausgenommen des Gruppengründers, können die Gruppe verlassen

FA90 Gruppenmitglieder können sich andere Gruppenmitglieder anzeigen lassen

3.1.3 Terminverwaltung

FA100 Jedes Mitglied einer Gruppe kann einen Termin erstellen

- Ersterfassung von Terminnamen, Terminzeit (in der Zukunft) und Terminort
- Visuelle Darstellung des gewählten Terminortes

WFA105 Der Terminersteller kann den Terminnamen, Terminzeit und Terminort nachträglich ändern

FA110 Der Gruppenmittelpunkt wird anhand der Standorte aller Teilnehmer berechnet

FA120 Jeder Termin wird jedem Gruppenmitglied angezeigt

FA130 Jedes Gruppenmitglied kann bei einem Termin zu- oder absagen

3.1.4 Terminablauf

FA140 Jeder Teilnehmer wird 30 Minuten vor Beginn des Termins benachrichtigt

WFA145 Teilnehmer werden auch bei geschlossener App benachrichtigt

FA150 Jeder Teilnehmer kann den Gruppenmittelpunkt ab 15 Minuten vor Beginn des Termins in einer Karte darstellen lassen

WFA155 Vom Gruppenmittelpunkt entfernte Teilgruppen erhalten einen eigenen Standort

WFA160 Teilnehmer können in der App anderen Teilnehmern mitteilen,

- dass sie bereits unterwegs sind

FA170 Der Termin löscht sich nach der Dauer des Termins (standardmäßig 60 Minuten)

3.1.5 Qualitative Anforderungen

Die Qualitativen Anforderungen beziehen sich auf unser Referenzgerät „Samsung Galaxy S4“(GT-I9505) mit Android 5.0.1

QA10 Die App soll auf jede Anfrage in durchschnittlich unter 5 Sekunden reagieren

QA20 Die App fährt in durchschnittlich unter 10 Sekunden hoch

QA30 Jede Funktion der App ist mit höchstens 5 Eingaben vom Startbildschirm der App zu erreichen

QA50 Unterstützt bis zu 50 Benutzer pro Gruppe

QA60 Jeder Benutzer kann in bis zu 20 Gruppen Mitglied sein

QA70 Der Server unterstützt das Anlegen von mindestens 30000 Benutzern

3.2 Nicht erfüllte Kriterien

Wir haben es nicht geschafft alle Wunsch-Kriterien zu implementieren. Diese können allerdings in zukünftigen Updates der goApp noch erfüllt werden.

3.2.1 Kontoverwaltung

WFA15 Der Benutzer kann zwischen Sprachen wählen

3.2.2 Gruppenverwaltung

WFA85 Der Gruppengründer kann seinen Status als Gruppengründer an ein Mitglied übergeben

WFA95 Gruppenmitglieder können die Teilnahmequoten anderer Gruppenmitglieder einsehen

3.2.3 Terminverwaltung

WFA105 Der Terminersteller kann den Termin löschen

3.2.4 Terminablauf

WFA160 Teilnehmer können in der App anderen Teilnehmern mitteilen,

- dass sie zu spät kommen
- dass sie schon am Terminort angekommen sind

WFA175 Die Dauer des Termins kann individuell festgelegt werden

3.2.5 Qualitative Anforderungen

QA40 Bei 99% aller Anwendungen werden Fehler abgefangen und führen nicht zum Absturz der App

4 Implementierungsplan

Während der Implementierungsphase haben sich einige Änderungen gegenüber unserem geplanten Verlauf ergeben. Zuerst hat jeder an seinem Teil der App auf dem Client bzw. Server gearbeitet. Dabei haben wir festgestellt, dass wir einige Änderungen brauchen -Abstimmung Client/Server -UnitTests -Google Verification 1. geplantes Diagramm 2. "richtiges"Diagramm

5 Unit-Tests

Zum Testen uneres Codes haben wir Unit-Tests benutzt. In dieser Phase prüfen wir nur die Funktionalität. Weitere Tests zu Fehlerfällen folgen in der Testphase.

5.1 Server

5.1.1 Datenbank

5.1.2 Algorithmus

5.1.3 Servlet

Zum Testen der Servlets werden die jeweiligen Management Klassen der Datenbank, sowie HTTP requests gemockt.

5.1.3.1 EventServlet

public void testCreate() Dieser Test prüft das erfolgreiche erstellen eines Events.

public void testGetParticipates() Dieser Test prüft das erfolgreiche Auslesen aller Teilnehmer der übergebenen Events.

public void testChange() Dieser Test prüft das erfolgreiche Ändern des Namens des Events.

5.1.3.2 GroupSearchServlet

public void testMemberSearch() Überprüft das erfolgreiche Einfordern einer Liste aller Gruppen, in denen ein gegebener Benutzer mitglied ist.

public void testNameSearch() Überprüft das erfolgreiche Abrufen einer Liste aller Gruppen, in deren Name die gegebene Suchanfrage enthalten ist.

5.1.3.3 LocationServlet

public void testSyncPos() Testet ob die gegebene Position des Benutzers in der Datenbank aktualisiert wird und der geclusterte Gruppenstandort des Events korrekt zurückgegeben wird.

5.1.3.4 LoginServlet Da jede der beiden Methoden auf den GoogleServer zum Verifizieren des GoogleTokens zugreift und man dies nicht testen kann, wird dieses Servlet erst manuell über den Client mit einem echten GoogleAccount getestet. Dort funktioniert das Registrieren, sowie auch Anmelden.

5.1.3.5 ParticipateServlet

5.1.3.6 RequestSearchServlet

public void testGetRequestsByGroup() Dieser Tests prüft das erfolgreiche holen aller Gruppen, zu dem von einem gegeben User, eine Anfrage existiert.

public void testGetRequestsByUser() Dieser Tests prüft das erfolgreiche holen aller Nutzer, zu dem von einem gegeben Gruppe, eine Anfrage existiert.

5.1.3.7 RequestServlet

public void testCreate() Dieser Test prüft das erfolgreiche Anlegen eines neuen Requests.

public void testAccept() Dieser Test prüft das erfolgreiche Akzeptieren eines Requests.

public void testReject() Dieser Test prüft das erfolgreiche Ablehnen eines Requests.

5.1.3.8 ServletUtils In den ServletUtils werden die 3 Methoden testIsUserAlreadyRegistrated(), testGetGoogleIdByToken() und testGetGoogleNameByToken() nicht getestet, da diese auf GoogleServer zugreifen und ein gültiges GoogleToken benötigen.

public void testCreateJSONParticipate() Dieser Test prüft das erfolgreiche Erstellen eines JSONObjektes aus einem Participate Objekt.

public void testCreateJSONListPart()

public void testCreateJSONEventID() Dieser Test prüft das erfolgreiche Erstellen eines JSONObjektes mit nur der EventID aus einem Event Objekt.

public void testCreateJSONEvent() Dieser Test prüft das erfolgreiche Erstellen eines JSONObjektes aus einem Event Objekt.

public void testCreateJSONLocation() Dieser Test prüft das erfolgreiche Erstellen eines JSONObjektes aus einem Location Objekt.

public void testCreateJSONGroup() Dieser Test prüft das erfolgreiche Erstellen eines JSONObjektes aus einem Group Objekt.

public void testCreateJSONUser() Dieser Test prüft das erfolgreiche Erstellen eines JSONObjektes aus einem User Objekt.

public void testCreateJSONListEvent()

public void testCreateJSONDoubleListEvent()

public void testCreateJSONListUsr()

public void testCreateJSONListGrp()

public void testCreateJSONGroupID() Dieser Test prüft das erfolgreiche Erstellen eines JSONObjektes mit nur der GroupID aus einem Group Objekt.

public void testCreateJSONListLoc()

public void testCreateJSONError() Dieser Test prüft das erfolgreiche Erstellen eines JSONObjektes aus einem ErrorCodes Objekt.

public void testExtractJSON()

5.1.3.9 UserServlet

public void testChangeName() Dieser Test überprüft das erfolgreiche Ändern des Namens des Nutzers.

5.2 Client

5.2.1 HTTPConnection

5.2.2 Model

5.2.3 View

5.2.4 Controller