Android-Go-App
Implementierungsbericht

Dennis Bäuml,
Theresa Heine,
Victoria Karl,
Tarek Wilkening

Betreuer:
Erik Burger,
Axel Busch,
Heiko Klare

19. Februar 2017

Inhaltsverzeichnis

1 Server

1.1 Implementierte Funktionen des Servers

/FS010/ Benutzerkonten verwalten

Über *RegistrationRequest* kann man ein neues Benutzerkonto auf dem Server anlegen. Das Benutzerkonto wird an die Geräte-ID des Smartphones gebunden und in der Datenbank des Servers hinterlegt.
Mit der *DeleteUserRequest* kann der zuvor erstellte Account gelöscht werden.

/FS170/ Umbenennen des Benutzers

Zusätzlich kann der Benutzer über die *RenameUserRequest* seinen Nutzernamen ändern.

/FS030/ Gruppen erstellen / löschen

Ein registrierter Benutzer kann via *CreateGroupRequest* Gruppen auf dem Server anlegen und per *DeleteGroupRequest* wieder löschen.
Der Benutzer ist automatisch in der eigens erstellten Gruppe Admin.

/FS040/ Gruppe: Benutzer einladen / entfernen

Mit der *CreateLinkRequest* kann ein Gruppenadmin einen Einladungslink erstellen. Das im Link enthaltene *'Secret'* wird in der Gruppe auf dem Server hinterlegt.
Über die *KickMemberRequest* kann ein Gruppenmitglied aus der Gruppe entfernt werden.

/FS180/ Gruppe: Mitglieder zu Administratoren machen

Mit der *MakeAdminRequest* können Mitglieder zu Admins befördert werden.

/FS060/ Gruppe: beitreten / verlassen

Ein anderer Benutzer kann mit einer *JoinGroupRequest* einer Gruppe beitreten.
Das zum Beitreten benutzer *'Secret'* wird aus der Gruppe vom Server gelöscht.
Eine Gruppe kann mit der *LeaveGroupRequest* verlassen werden.

/FS090/ Gruppe: Treffpunk festlegen

Mit der *SetAppointmentRequest* kann der Gruppenadmin einen Treffpunkt festlegen. Dieser wird in der Server-Datenbank gespeichert, wo er von anderen Gruppenmitgliedern abgerufen werden kann.

/FS110/ Gruppe: GPS Daten mitteilen / empfangen

Per *BroadcastGpsRequest* kann ein Gruppenmitglied der Gruppe seinen Standort mitteilen. Gleichzeitig erhält er, verpackt in einer *ObjectResponse*, die Standorte der anderen Gruppenmitglieder.

/FS190/ Gruppe: Umbenennen

Analog zur *RenameUserRequest*, kann man mit der *RenameGroupRequest*
Den Gruppennamen ändern (der Name muss eindeutig sein).

/FS130/ Verschlüsselung

Transportverschlüsselung wird über HTTPS garantiert.

1.2 Abweichungen vom Entwurf

1.2.1 Request / Response

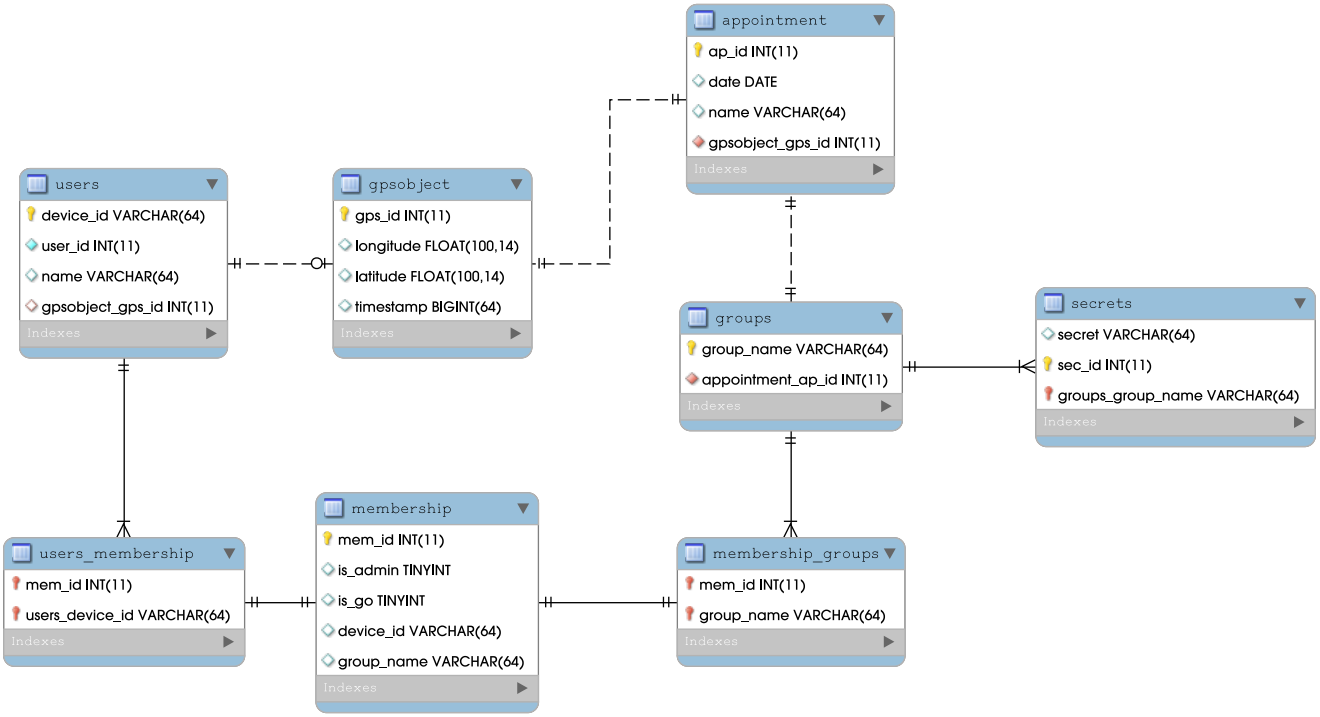
Da wird (leider erst nach der Entwurfsphase) festgestellt haben,
dass der Jackson ObjectMapper über Annotations auch Polymorphismus unterstützt,
ist ein Großteil der Logik in die Requests gewandert.
Um uns an das Command-pattern zu halten, haben wir und dazu entschieden,
den Requests eine execute() Methode hinzuzufügen. Somit sind Daten und Logik
zusammen gekapselt. Das Pattern erschien anfangs hinderlich, da man,
sobald sich das Model änderte, alle Requests anpassen musste.
Als das Model jedoch fertig war, wurde es sehr einfach, neue Requests für neue
Aufgaben hinzuzufügen.

Die Response Klassen waren, von Anfang bis Ende, nur Container für Parameter.
Zum Vereinfachen haben wir sie zu Response und ObjectResponse zusammengefasst.
In einer ObjectResponse können beliebige Objekte die Serializable implementieren,
verpackt werden.
Das Serializable Interface ist dabei nur für das Abbilden auf Subklassen zuständig,
die der ObjectMapper sonst nicht de-serialisieren könnte.
Die Requests sind immernoch grob in Group- und User-Requests unterteilt.

1.2.2 Servlets

Die Umstrukturierungen des Servers, hinsichtlich der Requests, haben dazu geführt, dass nur noch ein HttpServlet nötig war das MainServlet.
Es serialisiert/deserialisiert Request und Response Objekte bearbeitet Http-Anfragen.
Die GoApp kommuniziert ausschließlich über POST-Requests mit dem Server.
Über eine GET-Anfrage mit beispielsweise einem Webbrowser,
erhält man nur die Aufforderung, die App zu installieren.

1.3 Datenbank-Anbindung Server



Da unser ausgefallenes Gruppenmitglied der Meinung war, dass sich durch die Datenbank nichts an unserem Entwurf ändert, ist es leider auch nicht mit im Entwurf vorhanden. Da das Persistieren einer geschachtelten HashMap nicht ohne weiteres möglich ist, erschien es uns einfacher eine neue Klasse einzuführen, welche die Beziehung zwischen Benutzer und Gruppe modelliert. So entstand eine neue Tabelle Membership und die zugehörige Klasse MemberAssociation. Sie stellt eine ManyToMany Beziehung zwischen Gruppe und Benutzer dar.
OneToOne Beziehungen existieren ausserdem zwischen Group und Appointment, Appointment und GpsObject, sowie SimpleUser und GpsObject.
Die SSecrets"der Gruppen mussten als Collection ebenfalls in eine neue Tabelle ausgelagert werden.

Die Schnittstellen UserManager und GroupManager wurden zu ResourceManager zusammengelegt.
Diese zusätzliche Indirektion hat sich als nützlich erwiesen und macht die Datenbank zudem austauschbar.

Durch die MemberAssociations wurde auch das UserDecorator-Pattern überflüssig. Darin sind die Attribute gespeichert, die die Beziehung zwischen Benutzer und Gruppe beschreiben (Rechte, Go-Status etc.).