Fremdprojektanalyse

Projektbeschreibung

OneSocialWeb[1] ist auf XMPP aufsetzendes föderiertes Sozialesnetzwerk. Es erweitert XMPP um vCard-basierte Profile[2], Aktivity Streams[3], Benutzerrelationen und Zugriffskontrollisten und ermöglicht so Profile, Statusnachrichten, Medieninhalte, Artikel und viele andere Inhalte feingranular mit Personen zu teilen[4] und deckt so nach Fertigstellung alle von derzeitigen proprietären Sozialennetzwerken angebotene Funktionen ab, um diese letzendliche durch ein standardisiertes auf einer förderierten Architektur basiertes Soziales Netzwerk zu ersetzen.

Übersicht

Als XMPP basierter Dienst ist OneSocialWeb in Client und Server geteilt. Zur Zeit existieren ein vollständig (via Google Web Toolkit) mit JavaScript umgesetzter Web Client, ein textbasierter Konsolen Client und ein für das mobile Betriebssystem Android entwickelter Client. Die Serverkomponente von OneSocialWeb ist als Erweiterung des XMPP Servers Openfire realisiert.

Da alle Clients und auch der Server in Java implementiert sind, können Sie die gleichen Bibliotheken nutzen. Sowohl Client als auch Server nutzen dabei die Modellbibliothek, die im Wesentlichen ein gemeinsames Datenmodell für alle Komponenten bereitstellt und die für XMPP notwendige XML-Serialisierung und -Deserialierung durchführt. Die Client Bibliothek stellt eine für alle Clients einheitliche Schnittstelle für die, für das Projekt wichtigen Teile von XMPP, bietet und diese anhand des Datenmodells abstrahiert. Des Weiteren beinhaltet die Client Bibliothek eine auf der XMPP Bibliothek Smack[5] basierte Implementierung dieser Schnittstellen.

Schichtenaufteilung

In der Software lassen sich vier Schichten identifizieren: Präsentationsschicht, Client Schicht, Anwendungslogikschicht und Transportschicht, die sich jeweils zwischen Server und Client unterscheiden.

Die Präsentationsschicht ist für alle Clients unterschiedlich, besteht im Wesentlichen aber aus der Plattform und Benutzeroberflächen abhängigen Anzeige von Benutzeroberflächenkomponenten und der Weiterleitung von Ereignissen und Eingaben an die Client Schicht, die im

[1] `http://onesocialweb.org/`

[2] `https://tools.ietf.org/html/draft-ietf-vcarddav-vcardrev-19`

[3] `http://activitystrea.ms/`

[4] Nicht alle in den Activity Streams Spezifikationen beschriebenen Schemata werden auch in den Benutzerschnittstellen implementiert

[5] `http://www.igniterealtime.org/projects/smack/`

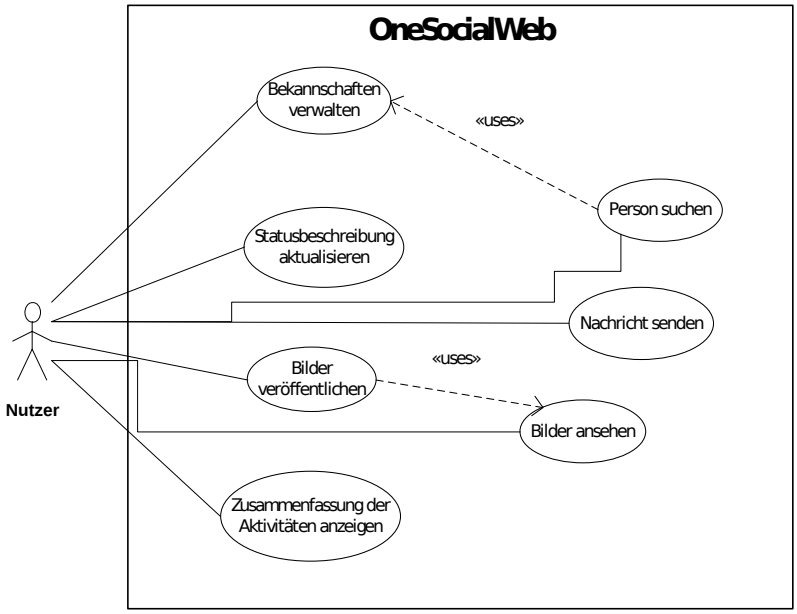

Abbildung 1: Use-Case-Diagramm

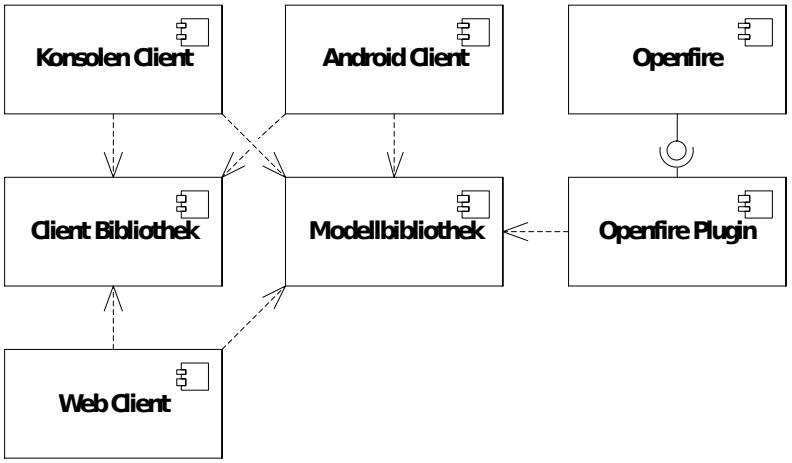

Abbildung 2: Komponenten von OneSocialWeb

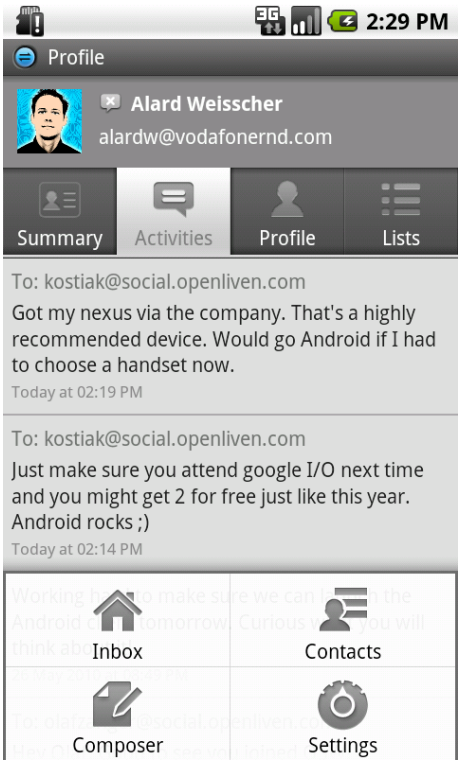

Abbildung 3: OneSocialWeb Android Client

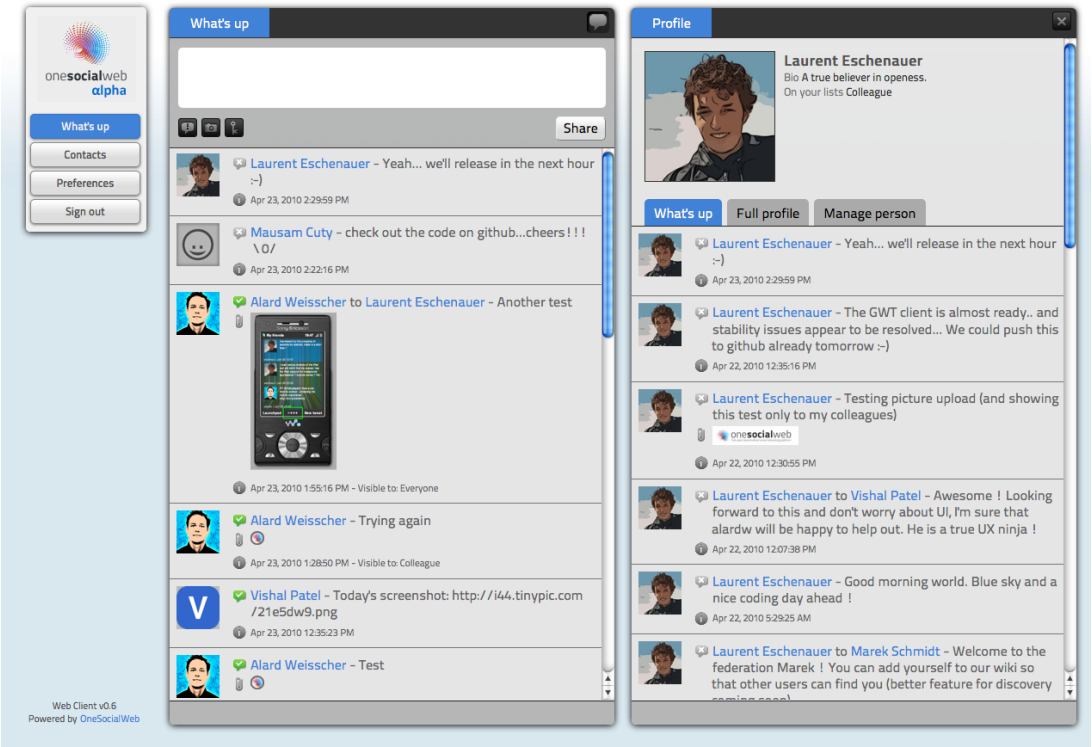

Abbildung 4: OneSocialWeb Web Client

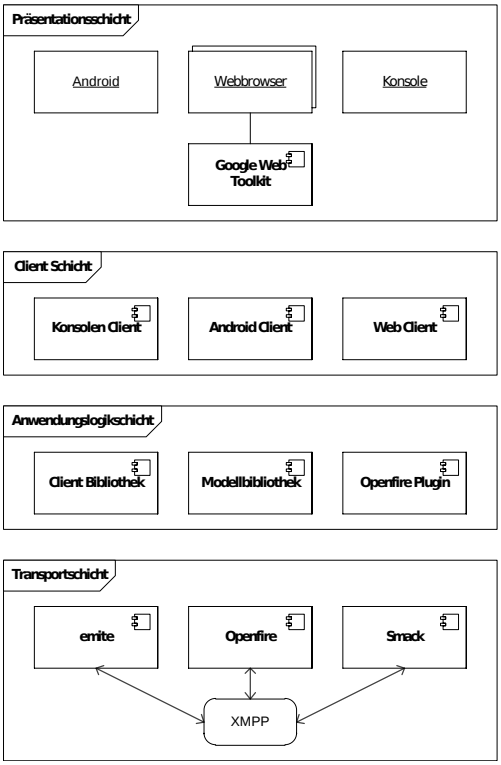

Abbildung 5: Schichten innerhalb von OneSocialWeb

Falle der minimalen und unvollständigen Administrationsoberfläche des Openfire Plugins entfällt, da diese Schichten hier nicht weiter getrennt sind.

Innerhalb der Client Schicht werden Eingaben und Ereignisse der Client Bibliothek, die zusammen mit der Modellbibliothek im Sinne einer Anwendungslogik die XMPP Kommunikation und Übertragung in das Datenmodell übernimmt, und der Präsentationssicht verarbeitet und Ergebnisse in Form von Protokoll-relevanten Ereignissen oder Daten oder Änderungen in der Benutzeroberfläche entsprechend in die Präsentations- und Anwendungslogikschicht weitergegeben.

In der Anwendungslogikschicht werden auf Client-Seite Ereignisse aus der Transport- und Client Schicht verarbeitet und mit Hilfe der Modellbibliothek in das OneSocialWeb zu Grunde liegende Datenmodell überführt oder daraus XMPP Nachrichten generiert Nachrichten generiert. Auf Server-Seite werden in der Anwendungslogikschicht mit Hilfe der Modellbibliothek und Openfire XMPP-Proktokoll Daten verarbeitet, verschickt und persistent gespeichert.

Umsetzung des Business-Delegate-Prinzips

Streng im Sinne des in Java EE umgesetzten Entwurfsmusters des Business-Delegates[6] wird das Business-Delegate-Prinzip in OneSocialWeb nicht umgesetzt.

In der Client und Modellbibliothek sind aber durch die den Einsatz des Entwurfsmusters der Fabrik der Aufspaltung in Schnittstelle und Implementierung weitestgehend getrennt und setzt so eine manuelle Form von Dependency-Injection um. So kann ein `OswService` für alle

[6] `https://java.sun.com/blueprints/corej2eepatterns/Patterns/BusinessDelegate.html`

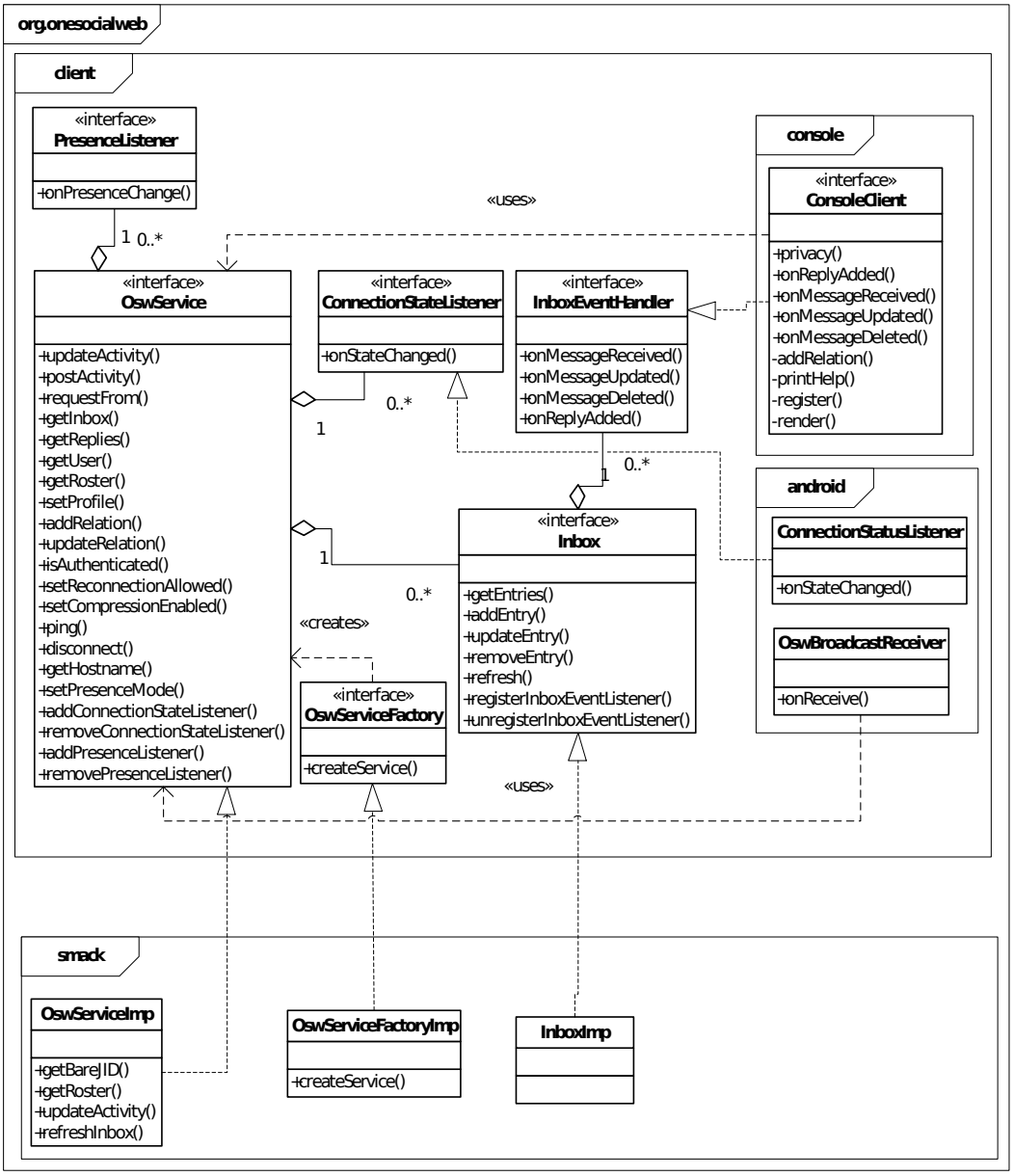

Abbildung 6: Klassendiagramm der Client Bibliothek und relevanter Client Klassen

Clients verschieden implementiert werden, ohne dass dies für die Client Bibliothek oder Teile des Clients außer der Client-spezifischen `OswServiceFactory` ersichtlich ist, so dass die entsprechenden Programmteile ohne Änderungen an anderen Teilen ausgetauscht werden können.

Alternativ können auch Client und Modellbibliothek als Business-Delegate für den XMPP-Server angesehen werden, wobei der DNS Resolver und eine Konfigurationsdatei als Lookup-Service dient. Client und XMPP-Server sind so (im gewissen Rahmen) unabhängig von einander, da der Client nur über im Datenmodell vorliegende Statusänderungen und Aktivitäten informiert und Kontakte und sein Profil ändern und auslesen lässt, ohne dass die Implementierung oder das Protokoll bekannt sein muss[7].

Aufbau der Benutzeroberfläche

Im Weiteren wird ausschließlich der Android Client betrachtet, da der Administrationsoberfläche des Servers, dem Konsolen und Web Client die zu identifizierenden Entwurfsmuster nicht oder nur teilweise auftreten.

Umsetzung des MVC Entwurfsmusters

Durch die in Android vorhandene Trennung von Konstruktion der Benutzeroberfläche, die deklarativ in XML-Dateien angegeben wird, ereignisverarbeitenden Actitivities wird ein dem passiven MVC Entwurfsmuster[8], in dem der Datenaustausch zwischen Model und View über den Controller erfolgt und z.B. in Cocoa verwendet wird, verfolgt, jedoch ist die Trennung schwächer, da Ereignisse in der Benutzeroberfläche direkt in einer Aktivität verarbeitet werden. Entsprechend der Architektur ist allgemein vorgesehen, dass längere und komplexe Aufgaben zu meist durch im Hintergrund ausgeführte Dienste abgewickelt werden. Dadurch hat die recht enge Kopplung zwischen Benutzeroberfläche und Activity kaum negative Auswirkungen. Darüber hinaus sind Activieties ohnehin Android-spezifisch wodurch Austauschbarkeit von View und Controller nicht möglich ist, so dass zur Portierung dieses Clients auf andere Plattformen eine ohnehin höhere Abstraktion gefunden werden müsste.

Bezogen auf den Android Client von OneSocialWeb wird auch genau diese Herangehensweise verfolgt: Actitivies verarbeiten Benutzereingaben und durch deinen im Hintergrund laufenden Dienst ausgelöste Ereignisse und passen dem entsprechend die Benutzerschnittstelle an, der Hintergrunddienst implementiert `OswService` und das Datenmodell wird durch die Modellbibliothek zur Verfügung gestellt.

Umsetzung des Action-Dispatch Konzepts und Ereignisketten

Das Action-Dispatch Entwurfsmuster, das zur Laufzeit einer Aktion einen Aktionsverarbeiter (Action-Dispatcher) zur Laufzeit ohne feste Bindung zu ordnet, wird in Android Anwendungen im weiteren Sinne durch Intents umgesetzt.

Intents dienen zur Laufzeit stattfindenden losen Bindung zwischen Aktivitäten, Diensten und Systemdiensten. Eine Komponente formuliert dabei einen Intent, der aus der auszuführenden Aktion bzw. das Ereignis, das aufgetreten ist, der Kategorie des Intents, die den se-

[7] Durch Verwendung von Datenstrukturen aus der Smack Bibliothek und Annahmen über das Transportprotokoll wird eine vollständige Implementierungs- und Transportschichtunabhängig zur Zeit aber noch nicht erreicht.

[8] `http://www.aspiringcraftsman.com/2007/08/25/interactive-application-architecture/`

mantischen Kontext, in der die Aktion ausgeführt werden soll, den Daten des Intents und sonstigen Metainformationen besteht und (sofern gewünscht) die Komponente an die der Intent gerichtet ist enthält, der dann von einer andere Komponente erfüllt oder wahrgenommen wird. Anwendungen können sich dabei für implizite Intents, bei denen keine Empfängerkomponente angegeben ist, registrieren. So können z.B. Systemereignisse, wie der Batteriestand oder Informationen zur Netzwerkverbindung, verteilt werden, Anwendungen aus anderen Anwendungen implizit anhand des Datentyps des Intents gestartet werden oder der Übergang zwischen Activities innerhalb der Anwendung gesteuert werden.

Innerhalb des OneSocialWeb Android Clients werden Intents zum Übergang zwischen Aktivities und zur Interaktion des Hintergrunddienstes mit dem Betriebssystem genutzt. Die Interkation zwischen dem Hintergrunddienst erfolgt jedoch wie von der Client Bibliothek vorgegeben gemäß dem Entwurfsmuster des Beobachters und die Bindung findet somit nicht zur Laufzeit statt.

Durch Intents und Nutzung des Entwurfsmusters des Beobachters lassen sich auch Ereignisketten aufbauen. So kann die `InboxActivity` beispielsweise, wenn noch kein Benutzername und Passwort eingestellt wurde, per Intent die `AccountSettings` Activity starten, die dann die Benutzername und Passwort an den `AndroidOswService` weitergibt, der dann die Verbindung aufbaut und schließlich indirekt über die Client Bibliothek über den `InboxAdapter` die `InboxActivity` über neue Aktivitäten informiert.