Hochschule für Telekommunikation Leipzig (FH)
Profilierung Mobile Applikationen

Dokumentation EtherApp

Thema: Entwicklung einer App zur Verwaltung von Etherpad Lite

Vorgelegt von: Ferdinand Malcher
Martin Stoffers

Betreuer: Prof. Dr. Ulf Schemmert

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol
UI	User Interface, Benutzeroberfläche
URL	Uniform Resource Locator

1 Systembeschreibung

1.1 Etherpad Lite

Etherpad Lite[1] ist ein kollaborativer Online-Editor. Anwender können Textdokumente, sogenannte Pads, gemeinschaftlich und in Echtzeit online bearbeiten. Die Oberfläche ist browserbasiert. Innerhalb der Pads ist durch farbliche Markierung gekennzeichnet, welche Bearbeitungen von welchem Nutzer vorgenommen wurden. Eine Instanz des Etherpad Lite ist standardmäßig offen. Damit kann jeder Anwender Pads anlegen und bearbeiten. Das System speichert Revisionen des Pad-Inhalts, die später abgerufen werden können. Die Oberfläche bietet außerdem die Möglichkeit zum Export des Inhalts in verschiedene Dokumentformate sowie einen Onlinechat zur Verständigung unter den Autoren.
Das Projekt ist Open-Source und damit quelloffen und frei für jeden.

1.2 Abgrenzung und Zielstellung

Die Oberfläche des Etherpad Lite ist schlicht und übersichtlich gehalten. Sie bietet benutzerrelevante Funktionen, die sich auf ein einzelnes Pad beziehen. Administrative Funktionen, die bestimmten Benutzerkreisen vorbehalten bleiben müssen, sind auf der Oberfläche nicht implementiert. Es gibt somit keine Möglichkeit, z.B. alle verfügbaren Pads anzuzeigen oder einzelne Pads zu löschen.
Es existiert eine umfangreiche HTTP-API, über die diverse Funktionen verfügbar gemacht werden. Für die praktische Verwendung dieser API ist ein Frontend sinnvoll.
Das Projekt EtherApp hat zum Ziel, einen Teil der API-Funktionen abzubilden und eine App zur Administration verschiedener Etherpad-Lite-Instanzen zu entwickeln. Das beinhaltet insbesondere folgende Funktionen:

- Anzeige aller Pads
- Anzeige von Meta-Informationen für einzelne Pads (Benutzer online, Revisionen, Datum, ...)
- Anlegen neuer Pads
- Löschen von Pads
- Anzeige des Pad-Inhalts
- Teilen der Pad-URL über soziale Dienste und E-Mail
- Rücksetzen des Inhalts auf ältere Revision
- Anzeige und Verwaltung von Gruppen
- Verwaltung mehrerer EPL-Instanzen und Profilverwaltung

2 Vorgehen

2.1 Zugriff auf die HTTP-API

Das EPL bietet von Haus aus eine umfangreiche HTTP-API[2]. Sie stellt eine Reihe von Funktionen für administrative Aufgaben bereit, die über die offene Weboberfläche nicht zur Verfügung stehen. Der Zugriff auf die API ist nur mit einem Shared Secret (API Key) möglich, der vom Administrator festgelegt werden muss.
Für den Zugriff auf die API existieren bereits Java-Bibliotheken, die alle API-Funktionen auf Methoden einer Java-Klasse abbilden. Für das Projekt EtherApp kam dabei die Implementation[3] des US-amerikanischen Entwicklers Jordan Hollinger zum Einsatz.
Die Bibliothek beinhaltet eine Klasse `EPLiteClient`, die mit den Zugangsdaten für die EPL-API initialisiert wird. Mit einem Objekt dieser Client-Klasse können Anfragen auf die jeweilige API durchgeführt werden, die als Java-Datenstruktur (`String` oder `HashMap`) zurückgegeben werden.

2.2 Selbst definierte Listen

Das Android-SDK stellt mit dem `ListView` ein View-Element zur Verfügung, mit dem Elemente in einer scrollbaren Liste angezeigt werden können.
Zur Anpassung von Daten und das tatsächliche Darstellen in der Liste wird ein Adapter benötigt. Android stellt bereits eine Reihe von Adaptern zur Verfügung (`BaseAdapter`, `ArrayAdapter`, ...). Deren Nachteil ist jedoch eine festgelegte Darstellungsweise; es ist auf einem Listenelement lediglich ein `TextView` dargestellt.
In unserem Anwendungsfall sollten in den Listenelementen der Padliste neben dem eigentlichen Pad-Namen zusätzliche Statusinformationen zum Pad sowie ein Löschen-Button angezeigt werden. Dieses Ziel konnte nur mit einem eigenen Adapter[1] und einem selbst gestalteten Listenelement[2] erreicht werden.

2.3 Asynchrones Laden der Listenelemente

Um eine komplette Padliste inklusive der Metadaten zu jedem Pad abzurufen, sind folgende Anfragen an die HTTP-API nötig:

- Abrufen der Padliste (nur IDs)
- Abrufen der Metadaten zu jedem einzelnen Pad
  - Anzahl der User online
  - Anzahl der Revisionen
  - Datum der letzten Bearbeitung

[1] `de.etherapp.adapters.PadlistBaseAdapter.java`
[2] `de.etherapp.beans.PadlistItem.java`

Für jedes Pad sind also drei HTTP-Anfragen abzusetzen. Bei einem Volumen von 300 Pads sind damit 901 HTTP-Anfragen nötig, um alle Daten der Padliste abzurufen, auch wenn der Benutzer diese zunächst gar nicht benötigt! Neben der dadurch ausgelösten Serverlast ist auch die resultierende Wartezeit für den Benutzer nicht akzeptabel. Dieser Umstand erfordert eine Lösung, mit der die Inhalte eines Listenelements erst dann geladen werden, wenn das Element auch angezeigt wird.
Dieses Ziel kann mit einem `AsyncTask` erreicht werden. Zunächst wird synchron die Liste aller Pads abgerufen und das `ListView` mit Elementen gefüllt. Jedes angezeigte Listenelement startet einen neuen Thread für jeden zu ladenden Wert. Der Thread stellt asynchron eine Anfrage an die API und schreibt die Ergebnisse direkt in die Views des Listenelements. Das asynchrone Laden ist beim Scrollen der Liste an der Zeitverzögerung zu erkennen.

Quellenverzeichnis

[1] Etherpad Foundation: *„Etherpad"*. `http://etherpad.org/` (23.01.2014).

[2] Etherpad Foundation: *„Etherpad v1.3.0 Manual & Documentation"*. `http://etherpad.org/doc/v1.3.0/#index_http_api` (23.01.2014).

[3] Hollinger J: *java-etherpad-lite*. `https://github.com/jhollinger/java-etherpad-lite` (23.01.2014).