
Distributed Computing

GK10.2 Mobile Dienste (BORM)

Systemtechnik Labor
5BHIT 2017/18

Pierre Rieger

Note:
Betreuer: Michael Borko

Version 1.0
Begonnen am 15. März 2018
Beendet am 9. April 2018

Inhaltsverzeichnis

1	Einführung	1
2	Ergebnisse	2
2.1	Der Server	2
2.1.1	Implementierung	2
2.2	Die App	3
2.2.1	Implementierung	3

1 Einführung

Es soll eine App erstellt werden, die in der Lage ist eine einfache Shoppingliste zu verwalten. Dabei sollen mehrere Clients zu einer Liste verbunden werden, dies sollen in der Lage sein Elemente einzufügen und zu Löschen.

2 Ergebnisse

2.1 Der Server

Der Server muss in der Lage sein mehrere Nutzer zu verwalten und diese einer Liste zuzuweisen. Um mehrere verschieden Listen zu verwalten wird jede Liste als Topic unterteilt. Jedes Topic besitzt einen eigenen Zustand der den derzeitigen Inhalt der Liste enthält. Darüber hinaus hat jedes Topic einen Namen und ein Passwort. Sobald der Nutzer einem Topic beitrifft erhält er den Zustand der derzeitigen Daten. Der Client kann dann Daten senden an den Server senden. Das könnte allerdings mit einem Algorithmus wie Diffsync noch performanter werden. Zurzeit verwendet der Server allerdings ein einfaches String Array.

2.1.1 Implementierung

Der Server ist als Node.js Server konzipiert, er verwendet Express und Websockets für die Kommunikation. Bei Websockets ist es wichtig viel Aufwand in Fehlerbehandlung zu legen. Ein Heartbeat Ping/Pong stellt sicher, dass keine 'toten Sockets existieren', Flow gibt Typsicherheit um gängige Programmierfehler zu vermeiden. In dieser Version können allerdings noch einige Fehler auftreten falls unerwartet Requests gemacht werden.

Der Server gliedert jede Einkaufsliste als einzelnes Topic, jedes Topic enthält eine Liste an Verbindungen und eine Liste für alle Daten die das Topic verwaltet. Ein Topic informiert alle Verbindungen sobald eine Veränderung auftritt und sendet die gesamte Datenliste. Ein weitere Verbesserung wäre es, wenn nur die Änderungen gesendet werden würden.

Der Server Antwortet im Datenformat JSON eine erfolgreiche notification schaut dann in etwa folgendermaßen aus:

```
{
  type: 'data',
  status: 'data',
  success: true,
  data: [ 'eier', 'milch', 'mehl' ],
  topic: 'topic'
}
```

2.2 Die App

Die App ist in React-Native entworfen und baut mittels Websockets eine Verbindung zum Server auf. Ein Client kann Topics auf dem Server erzeugen und ihnen beitreten, im Moment hat der Client kein 'Gedächtnis', er merkt sich also nicht in welchen Listen er bereits ist, darauf könnte in Zukunft aufgebaut werden indem der Client zum Beispiel RNFS (React-Native-File-System) verwendet um die Topic Namen zu speichern. Wenn der Client ein Topic erzeugen will dann sendet er eine JSON Request an den Server mit dem Wert **action: "new"** daraufhin bekommt er eine Rückmeldung ob die Erstellung erfolgreich war oder nicht. Im Moment werden misslungene Erstellungen nicht gehandhabt. Das Beitreten läuft ähnlich ab, genauso wie das Notifizieren.

2.2.1 Implementierung

Die Network Funktionalitäten werden getrennt vom GUI Code gehandhabt, das GUI bindet einzelne Listener an den Networker und initialisiert ihn sobald alle Funktionen gebunden sind. Der Networker ist ebenfalls als Singleton entworfen und bietet einige Funktionen um Daten, im vom Server erwarteten Format, zu senden. Beispielsweise werden die Daten am Server durch folgenden Code aktualisiert:

```
1  sendUpdateData(data : string[]) {  
    const name = this.name  
3    const pw = this.pw  
    if (!this.ws) return  
5    this.ws.send(JSON.stringify({  
        action: "notify",  
7        topic: name,  
        topicPass : pw,  
9        data: data  
    })))  
11 }
```

Listing 1: aktualisieren der Daten

Listings

1	aktualisieren der Daten	3
---	-----------------------------------	---