EIS

Entwicklung interaktiver Systeme

Sommersemester 2018

Implementationsdokumentation MS3

Team:

Jan Omar Mehr

Armin Weinrebe

Mentor:

Robert Gabriel

Dozenten:

Prof. Dr. Gerhard Hartmann

Prof. Dr. Kristian Fischer

Inhaltsverzeichnis

1 Allgemeine Voraussetzungen

1.1 Server

1.1.1 Node.js und NPM

Es werden die aktuellsten Version von Node.js (derzeit 9.11.1) und NPM (derzeit 6.1.0) benötigt um den Server ausführen zu können.
Download: `https://nodejs.org/en/download/` (beinhaltet NPM)

1.1.2 MongoDB

Für die Datenbank wird der aktuellste MongoDB Community Edition Treiber benötigt (derzeit 3.6.4).
Download: `https://www.mongodb.com/download-center?jmp=nav#community`
Die Datei mongod.exe muss ausgeführt werden und im Hintergrund laufen bevor der Server sich mit der Datenbank verknüpfen kann. Um die Daten besser visualisieren zu können, sollte zudem MongoDB Compass verwendet werden. So können auch die Backup-Daten sehr einfach importiert werden.
Download: `https://www.mongodb.com/download-center?jmp=nav#compass`

1.1.3 Command-Terminal

Um die Befehle von Node, NPM oder git ausführen zu können wird ein Terminal benötigt. Wir verwenden Git Bash für Windows.
Download: `https://git-scm.com/downloads`

1.1.4 REST-Client

Um den Server, ohne den in M2 folgenden Client, testen zu wollen, ist ein REST-Client nötig. Wir verwenden Insomnia.
Download: `https://insomnia.rest/download/`

1.1.5 Clone

Klone das gesamte Repository mit folgendem Befehl:

```
git clone https://github.com/jomehr/EIS_WeinrebeMehr.git
```

1.2 Client

1.2.1 Java

Es wird die Java Version 10 verwendet.

1.2.2 Android SDK

Um die Anwendung auf einem Android-Gerät installieren zu können, wird eine mind. SDK von 21 (aka Android 5.0 Lollipop) benötigt.
Die Ziel SDK ist 27 (aka Android 8.1 Oreo). Es wurde hauptsächlich auf Android 6 Marshmallow getestet.

1.2.3 Android Studio

Um die Anwendung auf einem echten Gerät oder einer VM installieren zu können, muss ein .apk-File erstellt werden. Dafür wird Android Studio benötigt. Download:`https://developer.android.com/studio/`

Nachdem Java und die benötigten SDK installiert wurden, kann auf dem verbunden mobilen Endgerät oder der VM, die Anwendung mit Android Studio gestartet werden (die benötigten dependencies werden automatisch installiert).

2 Installation

2.1 Server

2.1.1 Vorraussetzungen

Nodejs, NPM und MongoDB müssen installiert worden sein.

MongoDB muss im Hintergrund laufen (mongod.exe).

2.1.2 Installationschritte

Alle benötigten NPM-Module installieren

```
cd M3/Implementation/Server
```

- npm install

Server ausführen

- npm start

Je nach dem in welchem Netzwerk man sich befindet sind andere IP-Adressen (Variablen in www.js) nötig, um die Verbindung mit Clients zu ermöglichen.

- falls kein Netzwerk verwendet wird und der Client als VM auf dem Server läuft, nutze

```
const localHost = "127.0.0.1"
```

- falls ein echter Client im lokalen Wlan verwendet wird, nutze

```
const localWlan = "192.168.0.XXX
```

- falls ein echter Client mit mobilem Hotspot verwendet wird, nutze

```
const mobileHotspot = "192.168.48.XXX
```

Der Server sollte nun in Betrieb sein und in der Konsole bestätigen, dass

- er auf http://ip-adresse:port ansprechbar ist
- und sich erfolgreich mit MongoDB verknüpft hat

2.1.3 Nutzung

Der Server orientiert sich an der REST-Architektur und implementiert die HTTP CRUD-Methoden POST, GET, PATCH und DELETE

Die REST-Routen befinden sich hier:
`https://github.com/jomehr/SS18WeinrebeMehr/tree/master/M3/Implementation/Server/routes`

Die Schemas der Datenbank befinden sich hier:
`https://github.com/jomehr/SS18WeinrebeMehr/tree/master/M3/Implementation/Server/models`

Falls ein REST-Client verwendet wird, muss der body den Schemas entsprehchen, um die CRUD-Methoden erfolgreich durchführen zu können.
Dafür kann MongoDB Compass geöffnet werden.

- Falls noch keine Datenbank mit dem Namen 'db' vorhanden ist, muss diese zuerst erstellt werden.
- Dannach muss für jede Collection, die aus dem Ornder 'db-backups' importiert werden soll, eine leere Collection erstellt werden

- Diese sollte den gleichen Namen haben wie das zu importierende JSON
- Abschließend klicke, wenn du dich in der leeren Collection befindest, in der oberen Navigationsleiste auf 'Collection' und dann auf 'Import Data'
- Wähle die zugehörige JSON aus dem Dateipfad aus und klicke auf 'Import'
- Die Daten sollten nun erfolgreich importiert worden sein

Falls mit dem Android-Client getestet wird, müssen im Vorfeld Nutzer und eine Gruppe in der Datenbank vorhanden sein bzw. erstellt werden, da dies auf dem Client noch nicht möglich ist.

Kassenzettel können jedoch auf dem Client erstellt werden. Abrechnungen werden nach dem POST eines Kassenzettels automatisch erstellt.

2.2 Client

2.2.1 Vorraussetzungen

Die Java Version, die Android SDK und Android Studio müssen installiert worden sein.

2.2.2 Konfigurationsschritte

Je nachdem in welchem Netzwerk der Server läuft, muss die IP-Adresse in RestClient.java geändert werden. Es muss die gleiche IP-Adresse sein, auf welcher der Server läuft:

```
private static final String BASE URL = "http://XXX.XXX.XXX.XXX:1337/";
```

Falls nicht mit den DB Backup-Daten gearbeitet wird und neue Daten erstellt werden wollen, müssen in TabCreate.java bestimmte Zeilen angepasst werden:

- In Zeile 80 muss jedes Mitglied der Gruppe in das Array eingefügt werden:

```
dataUser.add(new UserData({UserId}, {UserName}, {GroupId));
```

- In Zeile 169 muss die UserId des derzeit „angemeldeten“ Nutzers eingefügt werden.
- In Zeile 188 muss die GroupId der Gruppe eingefügt werden:

```
receiptObject.put("group", {GroupId});
```

3 Abweichungen von Modellen

3.1 REST Modellierung

Der Server des vertikalen Prototypen Kassenzettel weicht in den Vorgaben der REST Modellierung ab. So sind für die Sub-Ressourcen der Receipt-Routen lediglich GET-Methoden weiter notwendig. POST-, PATCH- und DELETE-Methoden sind für den Client irrelevant geworden, da die im Kassenzettel referenzierten Objekte, bei seiner Erstellung, Änderung oder Löschung, automatisch mit verarbeitet werden. Auch der PATCH auf eine bestehende Abrechnung (/settlements/:settlementid) ist Teil des Automatisierungsprozesses und kann aus der Rest Modellierung entfernt werden.

3.2 Topic Modellierung

Auch von der Topic Modellierung wurde leicht abgewichen. Es werden die Erstellungen von Kassenzetteln, Gruppen und Abrechnungen gepublished, sowie eine neue Kassenzettel-Schuld, sobald sie entsteht. Bei der Erstellung einer Abrechnung wird diese nach der Topic Modellierung eindeutig spezifiziert. Im Prototypen wurde dies noch nicht umgesetzt. Zur Zeit würden somit jede Änderung an jeder Abrechnung für alle Nutzer gepublished werden. Für eine Veröffentlichung des Produkt müssten die Topics noch für Gruppen individualisiert werden.