Verteilte Systeme im Sommersemester 2023

Luca Schöneberg, Matr. Nr. 979049
Henry Wulff, Matr. Nr. 1014530 Osnabrück, 6.06.2023

Aufgabenblatt 1 - Websockets zur bidirektionalen Web-Client-/Server-Kommunikation

Anstelle der Umsetzung der CRUD-Operationen per http-Methoden sollen Sie Nachrichten definieren (JSON, XML, YAML,… oder eigenes Textformat), welche für den Datenaustausch zwischen Client und Server genutzt werden. Beschreiben Sie die Nachrichten im Protokoll zum Praktikum.

Websocket Endpoint für das Billboard.

Dieser Endpoint wird von der Klasse `BillBoardWebsocket` implementiert. Er ist für die Kommunikation zwischen dem Billboard und den Clients zuständig. Die Kommunikation erfolgt über JSON-Objekte. Die JSON-Objekte haben folgenden Aufbau:

```
{
"type": "messageType",
"content": "messageContent",
"sender": "messageSender"
}
```

type: Der Typ der Nachricht. Dieser kann folgende Werte annehmen:

- connection: Wird beim Verbindungsaufbau gesendet.
- set: Wird beim Setzen einer Nachricht gesendet.
- error: Wird bei einem Fehler gesendet.
- info: Wird bei einer Information gesendet.
- update: Wird bei einem Update gesendet.
- delete: Wird beim Löschen einer Nachricht gesendet.
- deleteAll: Wird beim Löschen aller Nachrichten gesendet.
- ping: Wird beim Senden eines Pings gesendet.
- pong: Wird beim Senden eines Pongs gesendet.
- close: Wird beim Schließen der Verbindung gesendet.
- unknown: Wird bei einer unbekannten Nachricht gesendet.

content: Der Inhalt der Nachricht. Dieser kann folgende Werte annehmen:

- string: Der Inhalt ist ein String.
- json: Der Inhalt ist ein JSON-Objekt.

sender: Der Sender der Nachricht. Dieser kann folgende Werte annehmen:

- server: Die Nachricht wurde vom Server gesendet.
- client: Die Nachricht wurde von einem Client gesendet.

Vergleichen Sie nach Abschluss der Realisierung das Ergebnis mit der Lösung aus Aufgabe 8:

1. Welche Unterschiede gibt es bzgl. der Adressierung von Objekten?

   Statische Objekte müssen genutzt werden, um sicherzustellen, dass alle Web-Socket-Instanzen auf dieselben Daten zugreifen können, während in REST-APIs jede Anforderung unabhängig von anderen behandelt wurde.

2. Welche Unterschiede gibt es im Hinblick auf die Identifikation der Kommunikationspartner?

   Bei RESTful-APIs wird die Identifizierung in der Regel über die IP-Adresse des anfragenden Clients vorgenommen. Bei WebSockets wird die Identifikation stattdessen über eine Session-ID realisiert, welche mit jeder Verbindung erstellt wird. Dies ermöglicht eine individuellere Kommunikation und ermöglicht es dem Server, Nachrichten speziell an einen Client zu senden.

3. Welches Verfahren ist effizienter im Hinblick auf die Verwendung von Netzwerk-Ressourcen? Rein theoretisch gesehen ist REST effizenter. Jedoch kann je nach anwendungs fall

   RESTful-APIs sind grundsätzlich effizienter, da sie eine Anforderungs-/Antwort-Struktur verwenden, was bedeutet, dass Netzwerkressourcen nur dann genutzt werden, wenn Daten angefordert oder gesendet werden. WebSockets hingegen halten eine offene Verbindung und können dadurch mehr Ressourcen beanspruchen. Allerdings erlauben WebSockets eine effizientere bidirektionale Kommunikation in Echtzeit-Anwendungsfällen, da Daten ohne zusätzliche HTTP-Anforderungen ausgetauscht werden können.

4. Aus Client-Sicht kann man in Web-Anwendungen Pull- und Push-basierte Zugriffsszenarien unterscheiden. Welche der im Praktikum eingesetzten Technologien (AJAX mit RESTful vs. Websockets) würden Sie vorrangig in welchem Szenario einsetzen?

   Im Falle von Pull-basierten Szenarien, bei denen der Client Daten anfordert, wenn er sie benötigt, sind RESTful-APIs (mit AJAX) ideal, da sie auf Anforderungs-/Antwort-Basis arbeiten. In Push-basierten Szenarien, bei denen der Server Daten an den Client sendet, wenn sie verfügbar sind, sind WebSockets vorzuziehen. WebSockets ermöglichen eine bidirektionale Kommunikation, was bedeutet, dass der Server Daten ohne eine vorherige Anforderung durch den Client senden kann. Dies ist insbesondere nützlich für Anwendungen, die Echtzeit-Daten benötigen, wie zum Beispiel Chat-Anwendungen oder Online-Spiele.