Chat App - Daten vom Server laden

Im nächsten Schritt ersetzen wir nun unsere Test-Daten mit den Daten vom Server. Wie genau der Server funktioniert ist für die Entwicklung des Frontends im Grunde genommen egal - Wir müssen nur wissen, wie wir an die Daten kommen und wie wir neue Daten hinzufügen können.
Fürs erste konzentrieren wir uns auf den ersten Schritt: Die Daten vom Server bekommen.

Als erstes müssen wir unserem Javascript ein par Module laden, die es uns erlauben, mit dem Server zu kommunizieren. Kopieren Sie dazu die folgenden `import` Befehle zu Oberst in Ihr JavaScript:

```
import { initializeApp } from "https://www.gstatic.com/firebasejs/9.20.0/
firebase-app.js";
import { getFirestore, orderBy, query, collection, doc, setDoc, onSnapshot,
addDoc, Timestamp } from "https://www.gstatic.com/firebasejs/9.20.0/firebase-
firestore.js";
```

Als nächstes müssen wir noch den den Kontakt zum Server etwas konfigurieren. Kopieren Sie dafür das folgende Dictionary in das JavaScript, direkt nach den Imports:

```
const firebaseConfig = {
            apiKey: "AIzaSyAQ9pCLKploVByyScEYnj3yMsoNFPIt14g",
            authDomain: "ffinfchat.firebaseapp.com",
            projectId: "ffinfchat",
            storageBucket: "ffinfchat.appspot.com",
            messagingSenderId: "787327787927",
            appId: "1:787327787927:web:a63c242f4ba509f53c1ad2"
        };
```

Nun können wir mit wenigen Befehlen den Kontakt zum Server starten:

```
const app = initializeApp(firebaseConfig);
const db = getFirestore();
```

```
// Eine Anfrage definieren
const q = query(collection(db, "messages"), orderBy("timestamp"));
```

Die `query` Methode definiert eine Anfrage an den Server. `db` ist die Datenbank. Wir sagen mit dem Aufruf, dass wir die Daten aus der Tabelle `messages`, geordnet nach Datum haben möchten.

Nun müssen wir die Anfrage noch absenden, das können wir mit der folgenden Methode machen:

```
const unsubscribe = onSnapshot(q, (querySnapshot) => {
            const messages = [];
            querySnapshot.forEach((doc) => {
                messages.push(doc.data());

            });
            updateComments(messages)
        });
```

Wir *abonnieren* damit die Datenbank. Das beutetet das wir ,wenn eine neue Nachricht auf dem Server eintrifft, sofort eine aktualisierte Liste mit Nachrichten bekommen. Wir erstellen eine leere Liste `messages` und laden sämtliche erhaltenen Nachrichten in die Liste.
Danach übergeben wir die Liste mit Nachrichten an die Methode `updateComments`, welche wir selber schreiben. Die Methode ist dann dafür verantwortlich die neuen Nachrichten auf der Webseite anzuzeigen. Das kann eigentlich gleich gemacht werden, wie mit unseren Testdaten von letzter Woche. Sie kriegen die Daten ausserm im genau selben Format wie sie bereits die Testdaten hatten.

#neufeld/ff