HTBL Imst

AUFBAULEHRGANG FÜR INFORMATIK

Notes App with Ionic (Angular) and Firebase

Name: Michael Bogensberger

Datum: 25.4.2020

1 Inhalt

2 Firebase

Firebase ist eine Entwicklungsplattform zur Entwicklung von mobilen und webbasierten Apps in der Google Cloud Plattform. Auf der Plattform wird eine breite Auswahl an Services zur Verfügung gestellt, welche die Entwicklung erleichtern.

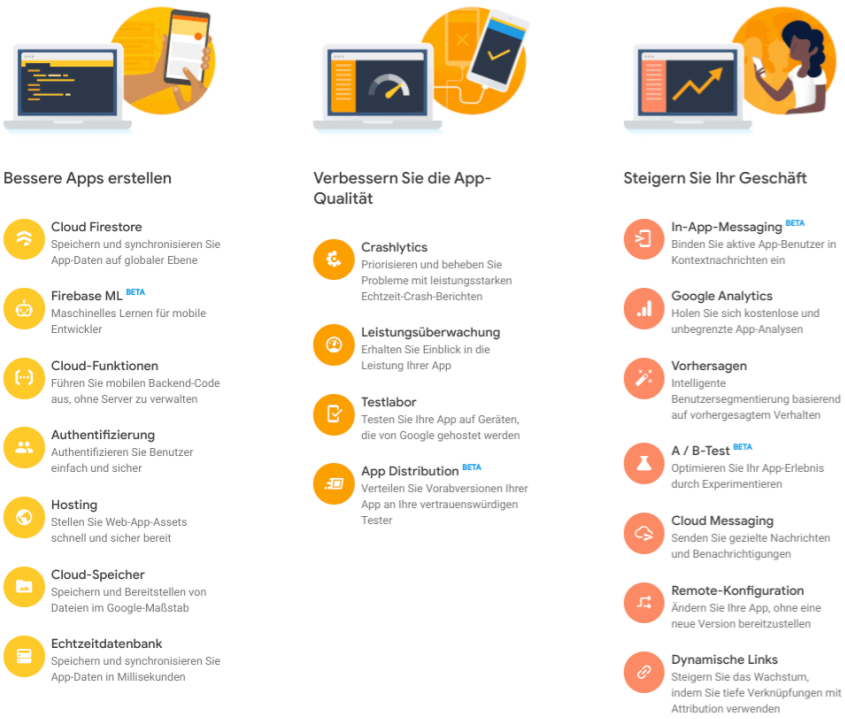

Abbildung 1 Firebase Angebot

In der Notes App werden wir Cloud Firestore verwenden. Cloud Firestore ist eine NoSQL-Datenbank mit der die Daten einer App global gespeichert, synchronisiert und abgefragt werden können. Die mobile- und Web-SDKs bieten die Einbindung der Daten in eine App, ohne einen eigenen Server einrichten zu müssen. Demnach lassen sich “True Serverless Applications” mit Firebase realisieren.

2.1 Projekt initialisieren

Zunächst müssen wir uns einen Account bei Firebase erstellen. Ist dies erledigt können wir auch schon ein Projekt erstellen. Dieses nennen wir hier einfach MyWebApp. Ist dies erledigt so sollten wir auf das Firebase Dashboard gelangen.

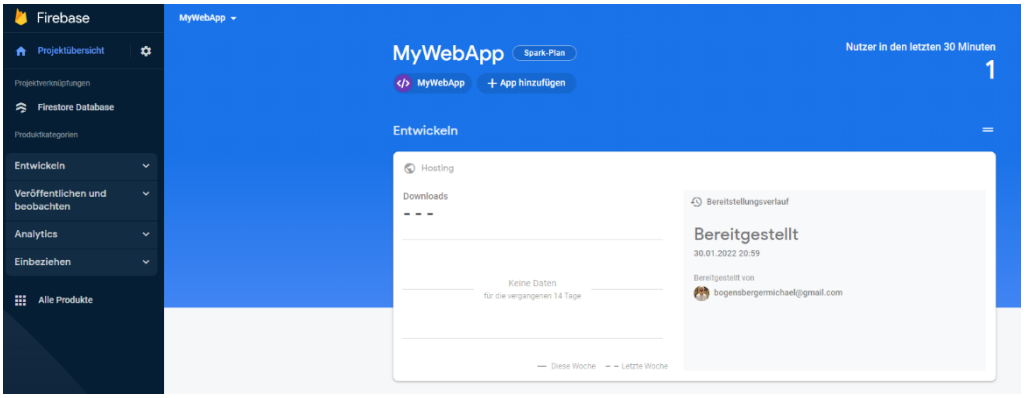

Abbildung 2 Firebase Dashboard

Nun müssen wir noch in die Einstellungen wechseln und unter Allgemein und Meine Apps gegbenfalls eine WebApp hinzufügen. Normalerweise sollte dies jedoch schon automatisch erstellt werden.

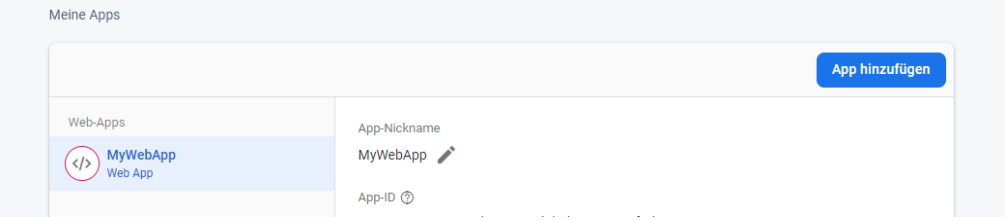

Abbildung 3 Firebase Settings

Nun müssen wir die Konstante `firebaseConfig` kopieren. Diese brauchen wir damit Firebase weiß welches Projekt gemeint ist. Den API Key von Firebase offen zu legt laut Google kein Sicherheitsrisiko da!

```
const firebaseConfig = {
  apiKey: "AIzASd345Ldaf2ASd6zV2AdrAVExaXg-qUKOU",
  authDomain: "mywebapp-d36c5.firebaseapp.com",
  projectId: "mywebapp-57617412",
  storageBucket: "mywebapp-d36c5.appspot.com",
  messagingSenderId: "64869952177",
  appId: "1:64869742398760:web:9e589b7aa7948sadv4712f9d",
  measurementId: "G-TJT80R6AAL"
};
```

Abbildung 4 Firebase Config

Zu guter Letzt müssen wir noch in Firebase auf die Firestore Database gehen und eine neue Datenbank erstellen. Wichtig ist hier noch, dass die Datenbank im Testmodus erstellt wird.

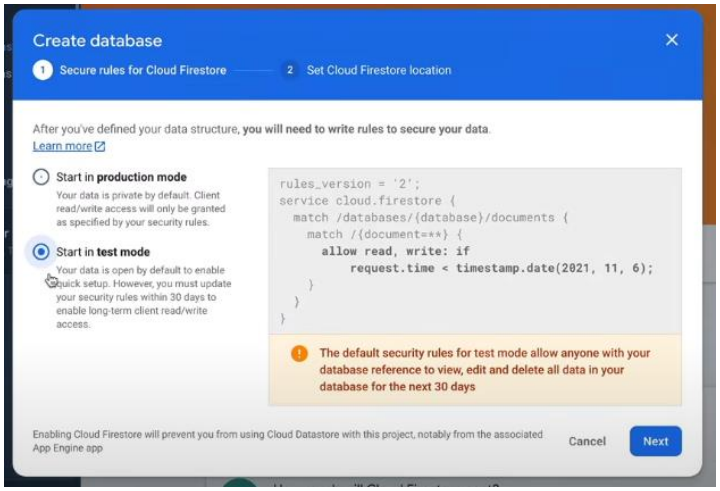

Abbildung 5 Firebase create Database

Unsere Datenbank sieht folgendermaßen aus. Wir haben eine Sammlung (Collection) mit dem Namen notes und darin befinden sich die jeweiligen Notizen. Es ist zu beachten das es sich bei Firestore um eine NoSql Datenbank handelt. Also um eine Dokumentbasierte Datenbank mit Collections.

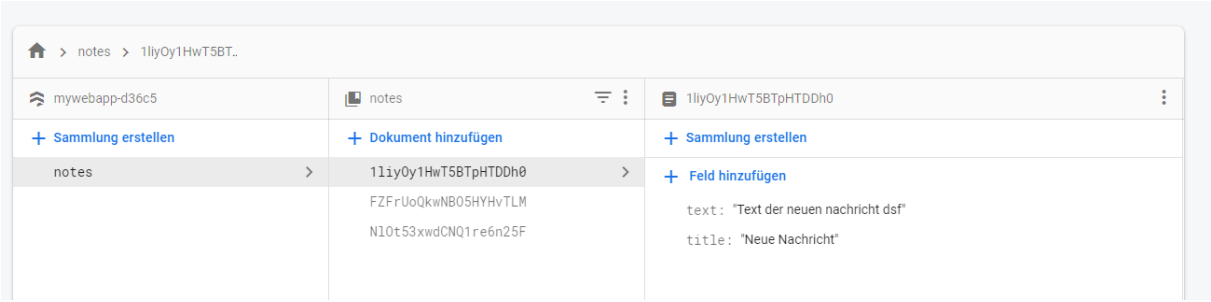

Abbildung 6 Firebase Database

3 Ionic

Ionic ist ein Open-Source-Webframework zur Erstellung von Hybrid-Apps und Progressive Web Apps auf Basis von HTML5, CSS, Sass und JavaScript/TypeScript. Progressive Web-Apps (PWAs) befinden sich in der Mitte zwischen WebView und nativen Apps. Dabei handelt es sich im Wesentlichen um kleine Websites, die in einer Browser-Shell innerhalb einer App laufen, die Zugriff auf die native Plattformschicht bietet. Die Entwicklung von plattformübergreifenden Frameworks wie Ionic, Flutter, React Native, Cordova und Xamarin bedeutet, dass PWAs inzwischen dem Verhalten und der Leistung der nativen Äquivalente ziemlich nahekommen können. Ionic-Apps werden in der Regel hauptsächlich in HTML, CSS und JavaScript geschrieben und das Framework ist mit den beliebten Javascript-Frameworks und -Bibliotheken wie Angular, Vue und React integriert.

Für uns ist nur wichtig, dass Ionic Components anbietet. Wie es in HTML zum Beispiel den tag <nav> gibt, gibt es in Ionic lauter Tags für eigene Komponenten wie zum Beispiel einer Card <ion-card>.

Guide zu Ionic: [Ionic Doku]

3.1 Init Ionic App

Zunächst führen wir folgenden Befehl aus, um die Ionic CLI auf unserem Gerät zu installieren.

- `npm install -g @ionic/cli`

Als nächstes erstellen wir uns mit folgendem Befehl ein neues Ionic Projekt. Falls gefragt wird ob wir React, Angular oder Vue verwenden wollen wählen wir Angular aus.

- `ionic start myWebApp blank`

Ionic bietet drei Grundlayouts an. Da wir keine Tabs oder eine Sidebar benötigen erstellen wir einfach eine Leere App.

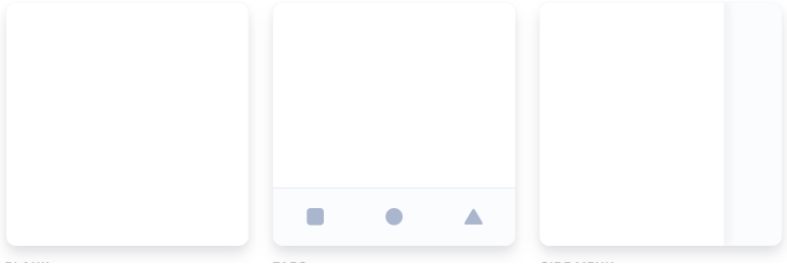

Abbildung 7 Ionic Layouts

Außerdem müssen wir noch AngularFire hinzufügen. AngularFire ist die Firebase JS SDK für Angular. Dazu führen wir folgenden Befehl aus:

- `ng add @angular/fire`
- Infos zu AngularFire:
  - Firebase Open Source Page
  - GitHub Repository zu Angular Fire

Nun müssen wir noch gegebenenfalls `npm install` ausführen, um alle Abhängigkeiten zu installieren. Ist dies erledigt können wir auch schon `ionic serve` ausführen und das Projekt wird gestartet.

4 Architektur

Die zuvor kopierte FirebaseConfig findet sich in der environments.ts Datei wieder. Des weiteren gibt es unter services einen DataService für die Logik der NotesApp. Sowol der DataService als auch das Frontend finden sich im app Ordner wieder. Wie wir im DataService sehen gibt es hier die CRUD Funktionen für die App. Zunächst injecten wir eine Referenz zu Firebase. Das ist für jeden Call nötig.

Wenn wir jetzt zum Beispiel getNotes() aufrufen wird die Konstante notesRef erstellt und collection aufgerufen. Dabei müssen wir collection die Referenz auf Firebase mitgeben und den Namen der Collection. Nun rufen wir collectionData auf und geben die Referenz notesRef mit. Mit dem idFiled sagen wir einfach, dass wir die ID mit zurückgeliefert haben wollen (ist dann das id field, weil wir es so genannt haben). Schlussendlich returnen wir die collectionData. Die anderen Funktionen funktionieren alle relativ gleich. Beim getNoteById() geben wir ein Observable zurück und suchen mithilfe einer id. Es ist außerdem zu beachten, dass Note ein Interface mit id, text und title ist.

```
export class DataService {
  constructor(private firestore: Firestore) {}

  getNotes() {
    const notesRef = collection(this.firestore, 'notes');
    return collectionData(notesRef, { idField: 'id' });
  }

  getNoteById(id): Observable<Note> {
    const noteDocRef = doc(this.firestore, `notes/${id}`);
    return docData(noteDocRef, { idField: 'id' }) as Observable<Note>;
  }

  addNote(note: Note) {
    const notesRef = collection(this.firestore, 'notes');
    return addDoc(notesRef, note);
  }

  deleteNote(note: Note) {
    const notesRef = doc(this.firestore, `notes/${note.id}`);
    return deleteDoc(notesRef);
  }

  updateNote(note: Note) {
    const noteDocRef = doc(this.firestore, `notes/${note.id}`);
    return updateDoc(noteDocRef, { title: note.title, text: note.text });
  }
}
```

Abbildung 8 Data Service

Hier sehen wir zum Beispiel das beim Button click auf der Startseite die Funktion addNote() aufgerufen wird.

```
<ion-fab vertical="bottom" horizontal="end" slot="fixed">
  <ion-fab-button (click)="addNote()">
    <ion-icon name="add"></ion-icon>
  </ion-fab-button>
</ion-fab>
```

Abbildung 9 addNote function

Hier werden alle Notizen ausgegeben. Hier machen wir einfach eine for each und geben jede Notiz aus.

```
<ion-card *ngFor="let note of notes" (click)="openNote(note)">
  <ion-card-header>
    <ion-card-title>{{note.title}} </ion-card-title>
  </ion-card-header>
  <ion-card-content>
    {{note.text}}
  </ion-card-content>
</ion-card>
```

Abbildung 10 jede Notiz ausgeben

```
async openNote(note) {
  const modal = await this.modalCtrl.create({
    component: ModalPage,
    componentProps: {id: note.id },
    breakpoints: [0, 0.5, 0.8],
    initialBreakpoint: 0.5
  });
  modal.present();
}
```

Abbildung 11 openNote Funktion

Wenn ich jetzt auf eine Notiz klicke, wird folgender Code ausgeführt. Hier wird das Modal was ein eigenständiger Component ist geöffnet und die ID mitgegeben.

In folgender Abbildung ist das Modal mit der einzelnen Notiz zu sehen. Wenn die Notiz existiert, wird das Modal angezeigt und der Titel und der Body ausgegeben. Beim Modal handelt es sich um einen eigenen Component. Jetzt kann ich die Notiz bearbeiten oder löschen.

```
<div *ngIf="note">
    <ion-card class="p-15">
      <ion-label position="stacked">Titel</ion-label>
      <ion-input [(ngModel)]="note.title"></ion-input>
    </ion-card>

    <ion-card class="p-15">
      <ion-label position="stacked">Titel</ion-label>
      <ion-textarea [(ngModel)]="note.text" rows="4"></ion-textarea>
    </ion-card>

    <ion-button class="m-10" expand="block" color="primary" (click)="updateNote()">
      <ion-icon name="save" slot="start"></ion-icon>
      Update
    </ion-button>

    <ion-button class="m-10" expand="block" color="danger" (click)="deleteNote()">
      <ion-icon name="trash" slot="start"></ion-icon>
      Deete
    </ion-button>
</div>
```

Abbildung 12 Modal

Die Dokumentation sollte soweit reichen um das Projekt beziehungsweise die Grundfunktionen von Firebase sowie Ionic zu verstehen. Der Code zum Projekt ist in der Abgabe beigelegt.

4.1 Fertige App

Gehen wir in den Browser und öffnen die App so sieht die App wie folgt aus. Ein weiterer toller Vorteil an Firebase ist, dass wenn sich in der Sekunde etwas in der Datenbank ändert, die Änderung direkt in der WebApp angezeigt wird. Man muss also nicht die Seite neu laden. Auch alle anderen Interaktionen benötigen kein neu laden der WebApp. Ist man auf der Startseite so sieht man alle Notizen. Klickt man nun auf eine der Notizen so öffnet sich ein Modal und wir können die Nachricht updaten oder löschen. Unten rechts findet sich ein Button, mit dem man eine neue Notiz hinzufügen kann.

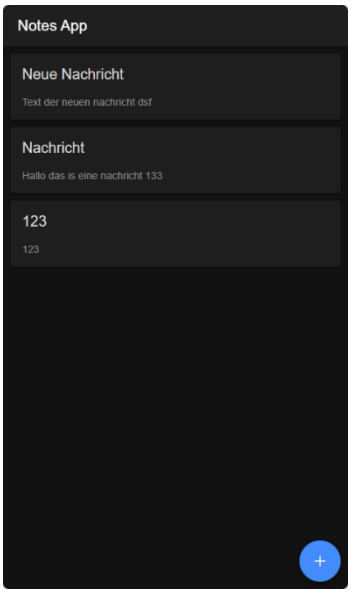

Abbildung 15 WebApp Startseite

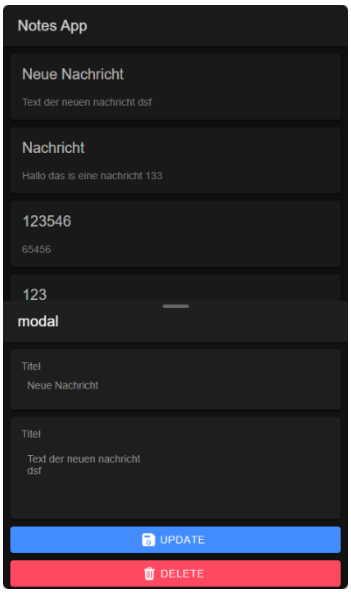

Abbildung 14 WebApp Modal

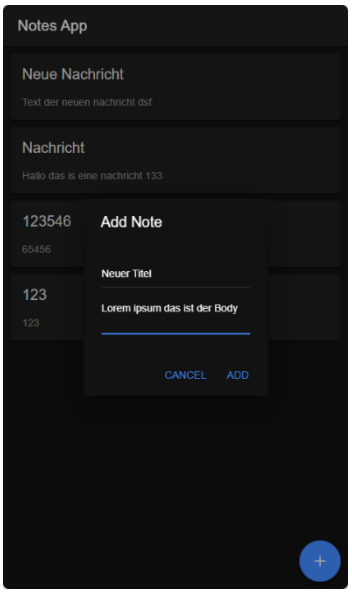

Abbildung 13 WebApp add Note

5 Abbildungsverzeichnis