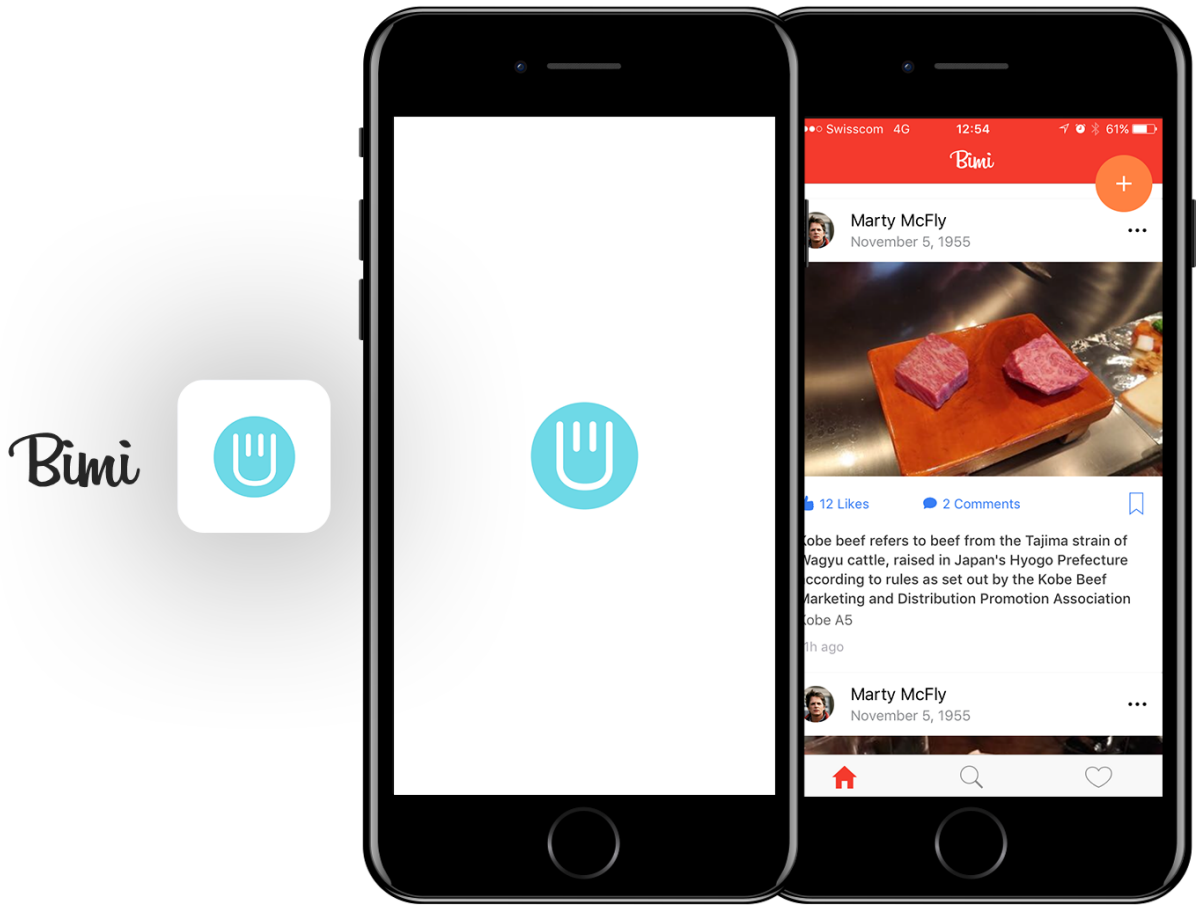
Aufgabe 3

Wir erstellen eine Food Instagram App und taufen sie Bimi (Japanisch für lecker!).

In der App verwenden wir das native Kamera Plugin. Wir verwenden die Kamera des Mobiltelefons um leckeres Essen zu Posten ;). Wir können entweder direkt ein Foto schiessen oder eines aus der Photolibrary auswählen und danach dem Bild noch einen Titel geben. Die Posts gehen auf deine Firebase Datenbank, wo wir die Daten persistieren und in unserem Feed darstellen.



1. Erstelle eine neue Ionic App im Tabs Design namens Bimi (solltest du aus den vorhergehenden Übungen noch kennen aber hier eine Starthilfe..)
    a. sudo ionic start Bimi tabs
    b. navigiere in das Projekt mit cd Bimi
    c. Füge noch die Android Plattform hinzu
    d. ionic platform add android
2. Installiere Angularfire2 damit wir einfach mit unserer Firebase DB kommunizieren können (im Root bzw. im Ordner indem du dich nach 1.d befindest).
    a. sudo npm install angularfire2 firebase --save
    b. Geh auf https://firebase.google.com/ und erstelle ein neues Projekt namens Bimi

c. Aktiviere den Authentifizierungsmodus damit anonymer Zugriff erlaubt ist. -> Sidemu auf Authenticatoin->SIGN-IN METHOD. Klicke auf Anonymous und setze den Status auf **Enable**.
d. Auch hier brauchen wir wieder die Config Inofs zu unsere Firebase DB. Auf dieser Authentication Seite oben rechts unter **WEB SETUP** findest du die Config. Kopiere sie für später.
e. Nun müssen wir noch die Regeln für Datenbankzugriffe ändern. Damit wir einfach los legen können definieren wir keine Regeln. Gehe dazu auf **Database** im Sidemenu und dann auf den Tab **RULES**. Aktualisiere den Eintrag mit folgendem Code

```
{
  "rules": {
    ".read": "auth == null",
    ".write": "auth == null"
  }
}
```

3. Zurück in den Code Editor! Wir importieren und initialisieren nun Firebase. Öffne die Datei src/app/app.module.ts und ergänze den Code. **Bitte verwende deinen Firebase Setttings damit du auch deine Daten in der Firebase DB siehst!**

```
...
//improt AngularFire NgModule
import { AngularFireModule } from 'angularfire2';
import { AngularFireDatabaseModule, AngularFireDatabase, FirebaseListObservable
} from 'angularfire2/database';
import { AngularFireAuthModule, AngularFireAuth } from 'angularfire2/auth';
import * as firebase from 'firebase/app';


export const firebaseConfig = {
    apiKey: "AIzaSyDI9hFwr2dTAUX97pEIFWR8CYxJJ",
    authDomain: "bimi-555.firebaseapp.com",
    databaseURL: "https://bimi-555.firebaseio.com",
    projectId: "bimi-555",
    storageBucket: "bimi-555.appspot.com",
    messagingSenderId: "461729483555"
};

@NgModule({
  declarations: [
    MyApp,
    AboutPage,
    ContactPage,
    HomePage,
    TabsPage,
],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp),
    AngularFireModule.initializeApp(firebaseConfig),
   AngularFireDatabaseModule,
   AngularFireAuthModule
  ],
...
```

4. In home.ts werden wir die Daten von Firebase ziehen um unsere Posts darzustellen. Gehe zu home.ts und importiere AngularFire und FirebaseListObservable

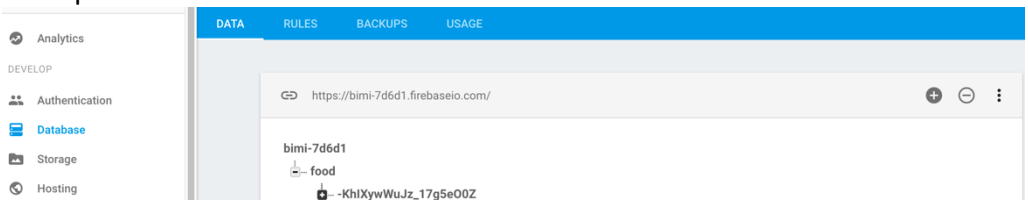
a.
```
import { AngularFireModule } from 'angularfire2';
import { AngularFireDatabaseModule,
AngularFireDatabase,FirebaseListObservable } from 'angularfire2/database';
import { AngularFireAuthModule, AngularFireAuth } from 'angularfire2/auth';
```

b. Dann brauchen wir noch die Member-Variablen
c. `photos: FirebaseListObservable<any>;`
d. `imageUrl: any;`
e. Im Construcotr füge AngularFire hinzu
f. `constructor(public navCtrl: NavController, public db: AngularFireDatabase, public afAuth: AngularFireAuth) {}`
g. Nun holen wir die Daten aus Firebase und speichern sie in der photos Variable
h. `this.photos = this.db.list('/food');`
i. Unsere Firebase DB ist noch leer darum erstellen wir kurz eine JSON Datei mit ein paar Beispieldaten und importieren sie in die Firebase Konsole direkt über den Browser im nächsten Schritt.

5. Erstellen ein Textfile mit folgendem Inhalt und speichere sie als data.json

```
{
   "food" : [
      {
            "title" : "Fast Food",
            "image" : "none",
         "name"  : "Burger"
      },
      {
         "title" : "Japaniese",
           "image" : "none",
         "name"  : "Sushi"
      },
      {
         "title" : "Snack",
           "image" : "none",
         "name"  : "Pizza"
      }
   ]
}
```

6. Geh auf Firebase und dort unter Database->DATA und klicke auf die drei Punkte und wähle Import JSON. Wähle deine data.JSON Datei aus.



7. Nun können wir im home.html weiterfahren und die Daten darstellen.

```
            <ion-card *ngFor="let photo of photos | async">
            <ion-item>
              <ion-avatar item-left>
                <img src="assets/img/marty-avatar.png">
              </ion-avatar>
              <h2>Marty McFly</h2>
              <p>November 5, 1955</p>
               <ion-icon item-right name="ios-more" tappable
(click)="deletePhoto(photo.$key)"></ion-icon>
            </ion-item>
            <img [src]="photo.image" />
            <ion-card-content>
              <p>{{photo.title}}</p><br>
              <p>{{photo.name}}</p>
            </ion-card-content>
            <ion-row>
              <ion-col>
                <button ion-button icon-left clear small>
                  <ion-icon name="thumbs-up"></ion-icon>
                  <div>12 Likes</div>
                </button>
```

```
        </ion-col>
        <ion-col>
          <button ion-button icon-left clear small
(click)="viewComments(photo.name)">
            <ion-icon name="text"></ion-icon>
            <div>4 Comments</div>
          </button>
        </ion-col>
        <ion-col center text-center>
          <ion-note>
            11h ago
          </ion-note>
        </ion-col>
      </ion-row>

    </ion-card>
```

8. Die siehst es fehlen noch einige Grafiken und in der Firebase DB haben wir für photo.image noch kein Bild sondern nur „none“ geschrieben. Das wird sich noch ändern. Zuerst holen wir mal das Bild von Marty. Auf dem Git findest du das bild marty-avatar.png. Kopiere dieses in dein Projekt unter src/assets/img/ (erstelle den Ordner fall nicht vorhanden) danach solltest du seinen Kopf sehen.
9. Nun installieren wir das Kamera Plugin! Dazu gebe folgende Consolen-Befehle im Rootverzeichnis deines Projektes ein
    a. `sudo ionic plugin add cordova-plugin-camera`
    b. `sudo npm install --save @ionic-native/camera`
10. Nun müssen wir noch das Plugin im Projekt importieren. Öffne app.module.ts und importieren die Camera
    a. `import { Camera } from '@ionic-native/camera';`
    b. Wichtig, füge die Camera auch zur Liste der Providers hinzu!

```
...
],
  providers: [
    StatusBar,
    SplashScreen,
    {provide: ErrorHandler, useClass: IonicErrorHandler},
    Camera
  ]...
```

11. So jetzt können wir die Kamera an einer beliebigen Stelle in der App aufrufen. Beliebig heisst, wir machen es in home.ts ;). Öffne home.ts und schreibe den Import
    a. `import { Camera, CameraOptions } from '@ionic-native/camera';`
    b. Im Constructor füge Camera hinzu
    c. `constructor(..., public camera: Camera){...}`
12. Nun können wir die Funktion schreiben um die Kamera aufzurufen. Nach dem Constructor schreibe die Funktion takePhoto()

```
takePhoto() {
    this.camera.getPicture({
      destinationType: this.camera.DestinationType.DATA_URL,
      targetHeight: 500,
      targetWidth: 500,
      correctOrientation: true,
      sourceType: 1 //0 = Photolibrary, 1 = Camera, 2 = Save to photoalbum
    }).then((imageData) => {
      this.imageUrl = ""+ imageData;
      this.openModal(this.imageUrl);
}, (err) => {
      console.log(err);
    });
  }
```

13. Mit sourceType kannst du angeben ob du die Kamera direkt aufrufen möchtest oder Bilder aus der Library wählen möchtest.

14. Wir brauchen jetzt noch die Funktion openModal(imageUrl). Hier rufen wir ein Modal auf und übergeben die Bild Url. Im Modal zeigen wir das soeben geschossene/ausgewähle Bild dar und der User kann hier noch einen Text eingeben. Danach über einen Send Button das ganze Posten!

```
openModal(imageUrl){
    let modal = this.modalCtr.create(Upload, {imgURL: imageUrl} );
    modal.present();
}
```

15. Jetzt wird einiges rot aufleuchten. Was wir brauchen ist eine Modal Page namens Upload und wir müssen den ModalController implementieren damit wir überhaupt ein Modal aufrufen (create) können. Also zuerst import des ModalController hier im home.ts
    a. import { ModalController } from 'ionic-angular';
    b. danach im Constructor hinzufügen
    c. constructor(..., public modalCtr: ModalController) {
    d. Nun die Modal Page namens Upload erstellen!
    e. sudo ionic g page Upload
    f. Bei Mac musst du evt. wieder die Schreib-/Leseberechtigung ändern.
    g. Nun gilt es wieder die neue Komponente in app.module.ts zu erfassen

```
import { UploadPage } from '../pages/upload/upload';
@NgModule({
  declarations: [
    MyApp,
    AboutPage,
    ContactPage,
    HomePage,
    TabsPage,
    UploadPage
],...
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    AboutPage,
    ContactPage,
    HomePage,
    TabsPage,
    UploadPage
  ],...
```

16. Und in home.ts ebenfalls importieren, da wir von hier auf diese Seite navigieren werden
    a. import { UploadPage } from '../upload/upload';
    b. Jetzt sollten keine Fehlermeldungen mehr auftauchen.
17. Nun brauchen wir noch einen Button um die Camera Funktion (takePhoto()) aufzurufen und dann vor dort auf unser Upload Modal zu gelangen. In home.html nach dem ion-content:

```
<ion-content>
  <ion-fab top right edge #fab>
    <button ion-fab color="danger"><ion-icon name="add"></ion-icon></button>
    <ion-fab-list>
      <button ion-fab (click)="takePhoto(fab)"><ion-icon name="md-camera"></ion-
icon></button>
      <button ion-fab (click)="takePhotoLibrary(fab)"><ion-icon name="md-
image"></ion-icon></button>
    </ion-fab-list>
  </ion-fab>
```

    a. Hier rufen wir die takePhoto() Funktion mit dem Parameter fab auf. Dies machen wir damit wir die einzelnen Listenpunkte des FABs wieder schliessen können. Also sobald wir ein Listenpunkt gedrückt haben sollen sie wieder

geschlossen werden. Wir müssen nun unsere takePhoto() Funktion in home.ts entsprechend anpassen.

```
takePhoto(fab: FabContainer) {...
fab.close();
```

b. Damit wir FabContainer benutzen können müssen wir diesen noch importieren. Füge bei den imports den FabContainer hinzu

```
import { FabContainer } from 'ionic-angular';
```

18. Nun kannst du noch die takePhotoLibrary(fab) funktion schreiben um Fotos von der Library auszuwählen. Tip: sourceType: 0;
19. Zum löschen eines Posts brauchen wir die Funktion deletePhotos()

```
deletePhoto(photoKey: string) {
 this.photos.remove(photoKey);
}
```

20. Nun können wir die Upload Page gestalten. Die Upload Page bekommt über den FAB bzw. die Kamera die URL des Bildes als Parameter -> in Schritt 14 mit let modal = this.modalCtr.create(UploadPage, {imgURL: imageUrl}); modal.present();
    a. Wir können nun in Upload den Navigationsparameter abfangen
    b. Im constructor von uploads.ts sollte public navParams: NavParams schon vorhanden sein. Falls nicht füge auch noch den `import { NavParams } from 'ionic-angular';` hinzu.
21. Nun brauchen wir ein paar Member Variablen um unsere Daten zu speichern. Füge folgende hinzu

```
 photos: FirebaseListObservable<any>;
form: any;
name: string = '';
imageUrl: any;
```

    a. importiere auch AngularFire und FirebaseListObservable damit wir das Bild und den Namen uploaden können

```
import { AngularFireModule } from 'angularfire2';
import { AngularFireDatabaseModule,
AngularFireDatabase,FirebaseListObservable } from 'angularfire2/database';
import { AngularFireAuthModule, AngularFireAuth } from 'angularfire2/auth';
```

Dann noch im Constructor `public db: AngularFireDatabase, public afAuth: AngularFireAuth`

22. Nun können wir die imageUrl welche wir von der Kamera als navParam bekommen haben abfangen. Im Construcotr können wir dies über mit navParams.get() machen.

```
this.imageUrl = navParams.get('imgURL');
```

23. Damit wir die Daten auf unsere Firebase DB pushen können brauchen wir noch

```
this.photos = this.db.list('/food');
```

24. Zum Senden des Posts brauchen wir noch die Funktion sendPost()

```
 sendPost(){
  this.photos.push({title: new Date().getTime(), image: this.imageUrl, name:
this.name});
  this.dismiss();
 }
```

du kannst für Title auch was anderes setzen, hier einfach als Beispiel ein Timestamp.
Auch rufen wir die Funktion dismiss() auf um unser Upload Modal zu schliessen. Definieren wir diese noch.

```
dismiss() {
  this.viewCtrl.dismiss();
}
```

Hier verwenden wir den ViewController um das Modal zu schliessen. Momentan wird this.viewCtrl.dismiss() noch rot unterstrichen. Wir müssen also zuerst noch den ViewController importieren.

```
import { ViewController } from 'ionic-angular';
```

und im Constructor hinzufügen

```
public viewCtrl: ViewController
```

25. Nun zur View upload.html. Wir erstellen hier ein Form damit wir kontrollieren können ob ein Text zum Bild und das Bild vorhanden sind, damit wir dann alles in die Firebase DB pushen können. Wir benutzen auch das Two-Way-Data-Binding z.B. mit [(ngModel)] ="imageUrl". Sprich wir bekommen das Bild von der Kamera über den NavParam und können ihn so direkt darstellen. Auch werden wir die Eingabe des Users vom Bildnamen so binden. Sobald beides zutrifft, also ein Text und ein Bild Vorhanden sind, blenden wir den Send Button ein.

```
<form
    [formGroup]="form"
    (ngSubmit)="sendPost()">

  <ion-item>
        <ion-label stacked>Food name:</ion-label>
        <ion-input
           type="text"
           formControlName="name"
           [(ngModel)]="name"></ion-input>
  </ion-item>

  <ion-item>
      <input type="hidden" name="image" formControlName="image"
[(ngModel)]="imageUrl">
      <img [src]="imageUrl" width="100%">
  </ion-item>
</form>
```

26. Wir brauchen noch den Send Button. Platziere den in der Navbar, auch füge noch ein Cancel Button hinzu um das Modal zu schliessen, falls der User sich entscheidet nichts zu Posten.

```
<ion-navbar>
  <ion-title>Upload</ion-title>
  <ion-buttons>
    <button ion-button (click)="dismiss()">
      <span showWhen="ios">Cancel</span>
      <ion-icon name="md-close" showWhen="android,windows"></ion-icon>
    </button>
  </ion-buttons>
  <ion-buttons end>
      <button ion-button icon-only (click)="sendPost()"
[disabled]="!form.valid">
      <ion-icon name="ios-paper-plane-outline"></ion-icon>
      </button>
  </ion-buttons>
</ion-navbar>
```

27. Nun müssen wir das Form noch kontrollieren bzw. validieren. Gehe dazu in upload.ts und importiere folgendes

```
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
```

im Constructor dann:

```
private _FB: FormBuilder
und im Constructor können wir jetzt prüfen
    this.form           = _FB.group({
         'name'         : ['', Validators.required],
         'image'        : ['', Validators.required]
      });
```

28. Jetzt kannst du die App Builden. Falls du eine Fehlermeldung „`Cant'resolve promise-polyfill`“ dann führe folgenden Befehl in der Konsole im Root deines Projektes aus:
    a. `npm install promise-polyfill --save-exact`

29. DONE !

30. Du kannst jetzt die App erweitern indem du z.B. alle geposteten Bilder in einer Explore / Search Page darstellst.
31. Etwas mehr Design könnte der App auch brauchen.

BONUS Content!

1. In home.html gibt es noch einen Funktionsaufruf namens viewComments(photo.name). Mit dieser Funktion gelangen wir auf eine Detail Page mit Kommentaren. Die Funktion sieht in home.ts folgendermassen aus.

```
viewComments(item){
  this.navCtrl.push(Comments, {
    item: item
  });
}
```

   Da Ionic neue Seiten via Push und Pop lädt pushen wir hier die Comments Page und übergeben den Parameter item mit dem Inhalt item, hier wäre es `photo.name`.
2. Wir brauchen noch die Comments Page. Erstelle die Comments Page
   a. `sudo ionic g page Comments`
3. importiere sie korrekt an den nötigen Stellen
   a. `home.ts`
   b. `app.module.ts`
4. Der Comments-Link sollte jetzt funktionieren.
5. Nun können wir den NavParam abfangen welcher wir von home.ts bekommen. In comments.ts erstelle die Member-Variable
   a. `commentsParam: any;`
   b. Im Constructor können wir nun den Parameterwert abfangen und in der soeben erstellten Variable speichern
   c. `this.commentsParam = navParams.get('item');`
6. Es fehlt noch die Anzeige des commentsParam in der View (comments.html)
   a. `<h2>{{commentsParam}}</h2>`
7. Done!