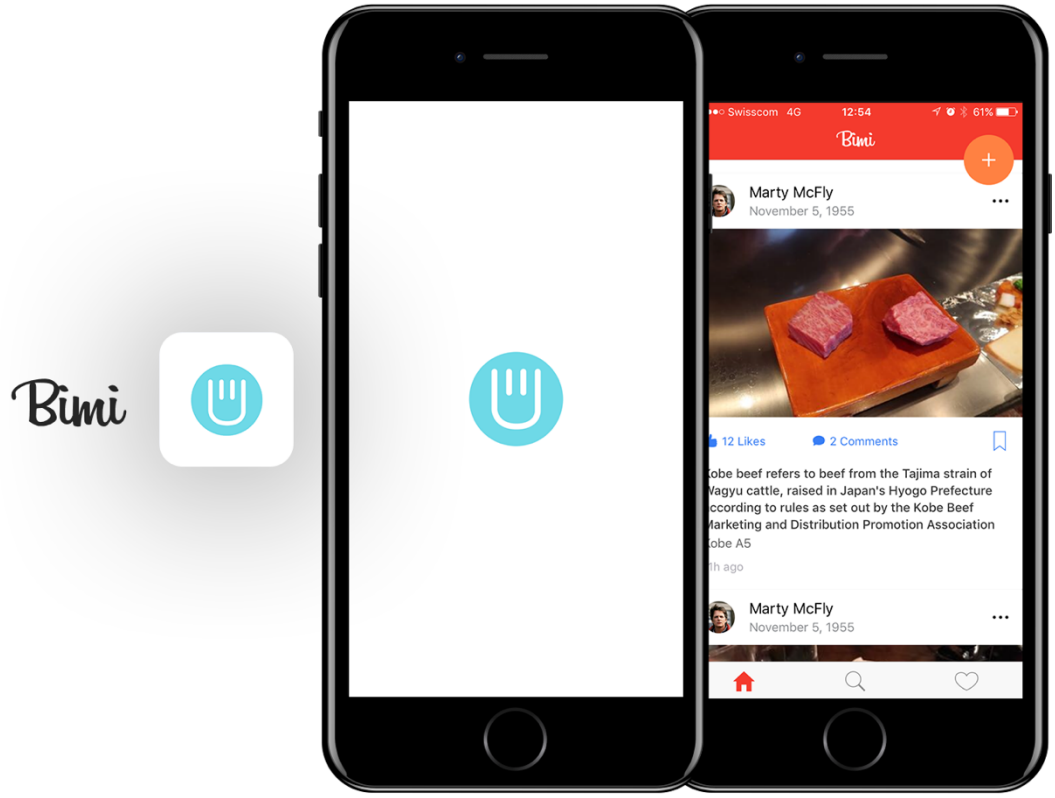
Aufgabe 4

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 Fotobibliothek 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.

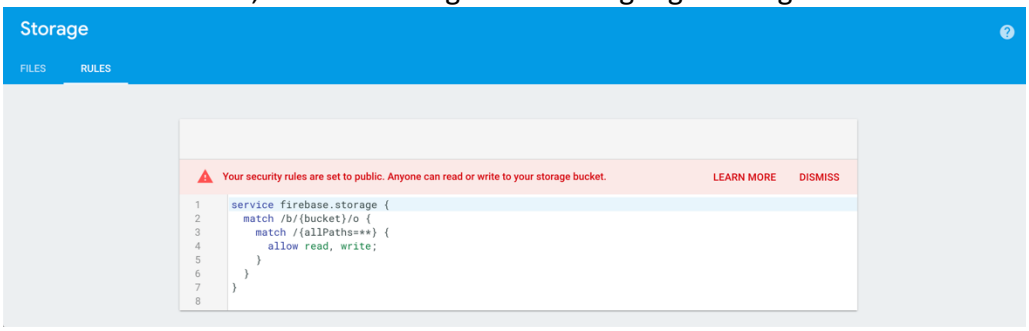


1. Erstelle eine Firebase Datenbank wie in Übung 3 (Schritt 1) und erstelle ein neues Ionic Projekt (Schritt 2-5 Übung 3). Installiere dabei auch firebase und angularfire2 (npm install firebase angularfire2 --save) und verknüpfe sie ebenfalls wie in Übung 3 in app.module.ts

```
import {AngularFireDatabaseModule} from 'angularfire2/database';
import {AngularFireModule} from 'angularfire2';
import { AngularFireStorageModule } from 'angularfire2/storage';


const firebaseConfig = {
  apiKey: "XXX",
  authDomain: "XXX",
  databaseURL: "XXX",
```

```
  projectId: "XXX",
  storageBucket: "XXX",
  messagingSenderId: "XXX"
};


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

2. Wir verwenden Firebase Storage um die Bilder zu speichern. Dazu solltest du in deinem Firebase Projekt unter DEVELOP > Storage einen Ordner namens **pictures** erstellen.
    a. Unter Rules, ändere die Zugriffsberechtigung wie folgt:



3. Nun haben wir unsere Firebase Datenbank und Fileablage erstellt. Als nächstes fügen wir in unserem Ionic Projekt in app.module.ts AngularFireStorage hinzu, damit wir nebst den Daten (Titel, URL Pfad zum Bild) auch die Bilder ablegen können.

In app.module.ts:

```
import { AngularFireStorageModule } from 'angularfire2/storage';
```

```
  imports: [
    ...
    AngularFireStorageModule,
```

4. Erstelle einen Provider namens firebaseService (ionic g provider firebaseService) und importiere darin:

```
import {AngularFireDatabase} from 'angularfire2/database';
import {AngularFireList} from 'angularfire2/database';
import { Observable } from 'rxjs/Observable';
```

5. Der Provider sollte dann folgendermassen aussehen

```
export class FirebaseServiceProvider {
```

```
  itemsRef: AngularFireList<any>;
  items: Observable<any[]>;

    constructor(public afd: AngularFireDatabase) {
    console.log('Hello FirebaseServiceProvider Provider');
    this.itemsRef = this.afd.list('/foodItems/');
    this.items = this.itemsRef.snapshotChanges().map(changes => {
      return changes.map(c => ({ key: c.payload.key, ...c.payload.val() }));
    });
  }

  getFoodItems(){
    return this.items;
  }

  addItem(newName, imgURL) {
    return this.itemsRef.push({ value: newName, imgUrl: imgURL });
  }



}
```

6. Als nächstes installieren wir das Camera Plugin, damit wir Fotos schiessen können oder Fotos aus der Fotobibliothek hochladen können. Du findest alle Ionic Plugins unter (https://ionicframework.com/docs/native/).
    a. npm install @ionic-native/core --save
    b. ionic cordova plugin add cordova-plugin-camera
    c. npm install --save @ionic-native/camera
    d. In app.module.ts müssen wir nun das Plugin noch importieren:

```
import { Camera } from '@ionic-native/camera';
```

    und bei den providers hinzufügen:

```
providers: [
    StatusBar,
    SplashScreen,
    Camera,
```

7. Nun können wir die Kamera in home.ts verwenden

```
import { Camera, CameraOptions } from '@ionic-native/camera';
```

und im Constructor injecten:

```
constructor(public navCtrl: NavController, private camera: Camera) {
```

8. Importieren wir hier noch unseren Provider damit wir später unsere Posts anzeigen können:

```
import { FirebaseServiceProvider } from '../../providers/firebase-service/firebase-service';
import { Observable } from 'rxjs/Observable';
```

    a. Und injecten diesen ebenfalls in unseren Constructor

```
  constructor(public navCtrl: NavController, public firebaseService: FirebaseServiceProvider, private camera: Camera) {
```

    b. Wir brauchen noch zwei Membervariabeln um das geschossene Bild zu speichern (imgURL) und eine um die Liste der Post zu speichern, welche wir von Firebase zurückbekommen (foodItems):

```
imgURL: any = "assets/imgs/default.jpg";
foodItems: Observable<any[]>;
```

c. Die Posts sortieren wir im Constructor, damit die neusten zuoberst erscheinen werden:

```
this.foodItems = this.firebaseService.getFoodItems().map((array) =>
array.reverse()) as Observable<any[]>;
```

9. Zum Ablauf eines Posts:
    a. Der User tippt auf einen FAB (Floating Action Button), wo er auswählen kann ob er ein Foto schiessen möchte, oder ein Foto aus der Fotobibliothek wählen möchte.
    b. Danach gelangt er auf eine neue Page wo das geschossene/gewählte Foto angezeigt wird und wo der User dem Bild noch einen Text anfügen muss.
    c. Wenn alles ausgefüllt ist, kann er das Bild posten und gelangt dann wieder auf den Home Screen.
10. Um den Ablauf in Schritt 9 zu erreichen, müssen wir zuerst noch eine Page für unseren Upload erstellen
    a. ionig g page upload
    b. Die Page in app.module.ts einbinden
    c. Die Page in home.ts einbinden
11. Jetzt können wir die Kamera aufrufen mit:

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

  try{
    const options: CameraOptions = {
      quality: 50,
      targetHeight: 600,
      targetWidth: 600,
      destinationType: this.camera.DestinationType.DATA_URL,
      encodingType: this.camera.EncodingType.JPEG,
      mediaType: this.camera.MediaType.PICTURE,
      correctOrientation: true
    }

    this.camera.getPicture(options).then((imageData) => {
      // base64 encoded string or a file URI:
      this.imgURL = '' + imageData;
      this.goToPage(this.imgURL);
      this.hideLoader();
     }, (err) => {
      // Handle error
      this.hideLoader();
    });
  }
  catch (e){
    console.log(e);
    this.hideLoader();
  }
```

```
}
```

a. Es wird noch einige Fehlermeldungen geben, da uns fab fehlt und displayLoader() sowie hideLoader() als auch goToPage().
Wir rufen hier fab.close() auf, damit der FAB nach Auswahl (Camera oder Bibliothek) wieder geschlossen wird. Dazu importiere in home.ts:

```
import { FabContainer } from 'ionic-angular/components/fab/fab-container';
```

b. Für den Loader importiere in homte.ts:

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

c. Definiere eine Membervariable:

```
loading: any;
```

d. Injecte den LoadingController in den Constructor:

```
constructor(public navCtrl: NavController, ..., public loadingCtrl: LoadingController) {
```

e. Schreibe die Funktionen:

```
  displayLoader(){
    this.loading = this.loadingCtrl.create({
       spinner: 'crescent',
       showBackdrop: true
    });
    this.loading.present();
  }

  hideLoader(){
    this.loading.dismiss();
  }

  goToPage(imgURL) {
    this.navCtrl.push(UploadPage, {
      imgURL: imgURL
    });
  }
```

12. Falls der User ein Bild aus der Bibliothek verwenden möchte:

```
takePhotoLibrary(fab: FabContainer) {
  fab.close();
  this.displayLoader();
  this.camera.getPicture({
    destinationType: this.camera.DestinationType.DATA_URL,
    targetHeight: 500,
    targetWidth: 500,
    correctOrientation: true,
    sourceType: 0 //0 = Photolibrary, 1 = Camera, 2 = Save to photoalbum
  }).then((imageData) => {
    this.imgURL = '' + imageData;
    this.goToPage(this.imgURL);
    this.hideLoader();
    this.content.scrollToTop();
  }, (err) => {
    console.log(err);
    this.hideLoader();
  });
}
```

a. Wir sehen hier beim sourceType:0 die Settings um auf die Bibliothek zuzugreifen.

13. Dein Constructor in home.ts sollte jetzt folgendermassen aussehen:

```
  constructor(public navCtrl: NavController, public firebaseService:
FirebaseServiceProvider,
     private camera: Camera, public loadingCtrl: LoadingController) {

    this.displayLoader();

    //order array in reverse
    this.foodItems = this.firebaseService.getFoodItems().map((array) =>
array.reverse()) as Observable<any[]>;

    //hide loader
    this.foodItems.subscribe(x => this.hideLoader());


  }
```

a. Beim öffnen der App, zeigen wir ebenfalls einen Loader an, bis die Daten von Firebase ankommen.

14. Die home.html View:

```
<ion-header>
  <ion-navbar>
    <ion-title>Bimi</ion-title>
  </ion-navbar>
</ion-header>

<ion-content style="background-color:#efeff6">

    <ion-fab right bottom #fab>
        <button ion-fab color="primary"><ion-icon name="add"></ion-
icon></button>
        <ion-fab-list side="top">
        <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>


  <ion-card *ngFor="let item of foodItems | async">
    <ion-item>
      <ion-avatar item-left>
        <img src="assets/imgs/marty-avatar.png">
      </ion-avatar>
      <h2>Marty McFly</h2> <p>November 5, 1955</p>
      <ion-icon item-right name="ios-more" tappable
(click)="removeItem(item.key)"></ion-icon>
    </ion-item>
    <img src="{{item.imgUrl}}" onError="'assets/imgs/default.jpg'" />
    <ion-card-content style="text-align:left">
```

```
      <p>{{item.value}}</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>
          <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>
</ion-content>
```

15. Kommen wir zum Upload
    a. Wir bekommen hier das Bild welches wir in Home erfasst haben und Validieren mit einem Formular ob Bild und ein Text vorhanden ist. Erst dann kann der User den Post senden.
    b. Unsere Imports sehen folgendermassen aus:

```
import { FirebaseServiceProvider } from '../../providers/firebase-
service/firebase-service';

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

import {storage} from 'firebase';

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

    c. Wir definieren folgende Membervariabeln:

```
  form: any;
  name: string = '';
  imgUrl: any;
  loading: any;
```

    d. Unser Constructor:

```
  constructor(public navCtrl: NavController, public navParams: NavParams,
public firebaseService: FirebaseServiceProvider, private _FB:
FormBuilder, public loadingCtrl: LoadingController) {

    console.log('imgURL', navParams.get('imgURL'));
    this.imgUrl = navParams.get('imgURL');


    this.form = _FB.group({
    'name' : ['', Validators.required],
```

```
    'image' : ['', Validators.required]
    });

  }
```

e. Im Constructor setzen wir die Validierung des Forms. In der View können wir auf ihre Validität prüfen und den Send Button aktivieren/deaktivieren. Dazu später in Schritt d.

f. Auch hier verwenden wir den Loader und brauchen:

```
  displayLoader(){
     this.loading = this.loadingCtrl.create({
        spinner: 'crescent',
        content: 'Posting...'
     });

     this.loading.present();
  }

  hideLoader(){
     this.loading.dismiss();
  }
```

g. Zum Schluss noch unsere Post Funktion:

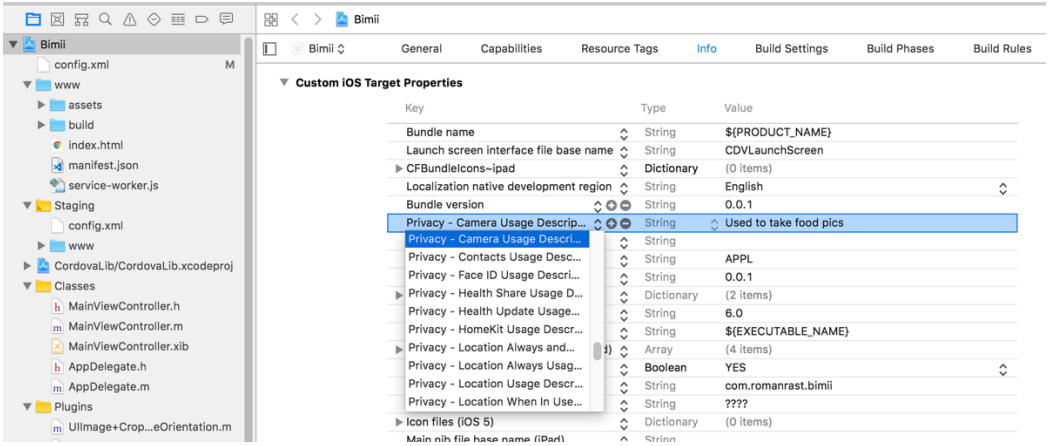
```
  sendPost(){
    this.displayLoader();
    const file = this.imgUrl;
    const picutres = storage().ref('pictures/'+ 'userID' + new
Date().getTime()+'.jpg');
    picutres.putString(file, 'data_url').then((snapshot)=>{
      let imgURL: any = snapshot.downloadURL;

      this.firebaseService.addItem(this.name, imgURL).then((data)=>{
        this.navCtrl.pop();
        this.hideLoader();
      });

    });

  }
```

h. Nun zur View upload.html

```
<ion-header>

  <ion-navbar>
    <ion-title>upload</ion-title>
    <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>

</ion-header>
```

```
<ion-content padding>


<form [formGroup]="form" (ngSubmit)="sendPost()">
  <ion-grid>
    <ion-row>
      <ion-col col-9>
        <ion-item>
          <ion-label stacked>Food name:</ion-label>
          <ion-input type="text" formControlName="name"
[(ngModel)]="name"></ion-input>
        </ion-item>
      </ion-col>
      <ion-col col-3>
        <ion-item>
          <input type="hidden" name="image" formControlName="image"
[(ngModel)]="imgUrl">
          <img [src]="imgUrl" width="100%">
        </ion-item>
      </ion-col>
    </ion-row>
  </ion-grid>
</form>
</ion-content>
```

16. Die App sollte nun funktionieren!
17. WICHTIG auf iOS ab iOS 10 müssen wir bei gebrauch der Kamera, dem User bezüglich Privacy Bestimmungen noch sagen, warum diese verwendet wird.
    a. Dazu builde deine App mit ionic cordova build ios
    b. Öffne das Projekt unter Bimi/platforms/ios/Bimi.xcodeproj
    c. Wähle das Projekt an und unter Info auf einer Zeile auf das + Icon tippen und **Privacy – Camera Usage** Description auswählen.
    d. Bei Value einen Text eingeben, warum die Kamera verwendet wird, wie folgt:



18. Jetzt kannst du die App noch Stylen!
19. Done 🎉