Aufgabe 2

Wir erstellen eine App **Rhyschwimme**, welche Daten durch einen GET-Request erhält. Die Daten werden im JSON Format geliefert und wir bereiten sie auf, damit wir die Wassertemperatur des Rheins in Basel schön anzeigen können. Mit dieser Übung lernst du wie man einen Shared Service (der Service kann in jeder Page verwendet werden) erstellt und etwas über das Stylen der App.

1. sudo ionic generate provider WaterData
2. sollte so aussehen

```
import { Injectable } from '@angular/core';
import { Http } from '@angular/http';
import 'rxjs/add/operator/map';

/*
  Generated class for the WaterData provider.

  See https://angular.io/docs/ts/latest/guide/dependency-injection.html
  for more info on providers and Angular 2 DI.
*/
@Injectable()
export class WaterData {

  constructor(public http: Http) {
    console.log('Hello WaterData Provider');
  }

}
```

- Info: Rxjs brauchen wir für den map operator welcher uns erlaubt mit dem Observable zu arbeiten, welcher wir vom Http Server zurück bekommen.

3. Füge den Provider zur Liste aller Providers in app.module.ts hinzu

```
import { NgModule, ErrorHandler } from '@angular/core';
import { IonicApp, IonicModule, IonicErrorHandler } from 'ionic-angular';
import { MyApp } from './app.component';
import { Page1 } from '../pages/page1/page1';
import { Page2 } from '../pages/page2/page2';

import { WaterData } from '../providers/water-data';

import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';

@NgModule({
 declarations: [
  MyApp,
  Page1,
  Page2
 ],
 imports: [
  IonicModule.forRoot(MyApp)
 ],
 bootstrap: [IonicApp],
 entryComponents: [
  MyApp,
  Page1,
  Page2
 ],
 providers: [
  StatusBar,
  SplashScreen,
  {provide: ErrorHandler, useClass: IonicErrorHandler},
  WaterData
 ]
})
export class AppModule {}
```

4. Füge zwei Member Variable zu water-data.ts hinzu, apiUrl und data. Setze für die apiUrl, die Url zu den JSON Daten für die Wassertemperatur ein. Den typ für die Variable data kannst du als „any“ setzen. Danach schreibe die Funktion getWaterData() mit dem GET-Call auf die apiUrl.

```
@Injectable()
export class WaterData {

 //https://developer.yahoo.com/yql/
```

```
 apiUrl =
"https://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20csv%20wher
e%20url%3D'http%3A%2F%2Fwww.hydrodaten.admin.ch%2Fgraphs%2F2613%2Fte
mperature_2613.csv'&format=json&env=store%3A%2F%2Fdatatables.org%2Falltable
swithkeys";
 data: any;


 constructor(public http: Http) {
  console.log('Hello WaterData Provider');
 }

getWaterData(){
   return new Promise(resolve => {
    this.http.get(this.apiUrl)
    .map(res => res.json())
    .subscribe(data => {
     console.log(data);
     //return data;
     this.data = data;
     resolve (this.data);
         });
   });

 }

}
```

5. Nun kannst du den Provider in jeder Page benutzen indem du ihn wie folgt importierst:
   import { WaterData } from '../../providers/water-data';
6. Danach füge den Provider zu den @Components deiner page1.ts hinzu.
   ```
   @Component({
    selector: 'page-page1',
    templateUrl: 'page1.html',
    providers: [WaterData]
   })
   ```
7. Als nächstes, füge den WaterData Provider/Service deinem Constructor hinzu
   ```
   constructor(public navCtrl: NavController, public waterService: WaterData) {
    }
   ```
8. Füge zwei weitere Member Variablen waterData und waterTemp vom typ any hinzu
   ```
   ...export class Page1 {
    waterData: any;
    waterTemp: any;
    constructor(public navCtrl: NavController, public waterService: WaterData) {...
   ```
9. Füge den Service-Call nach dem Constructor hinzu:
   ```
   getWaterData(){
    this.waterService.getWaterData()
   ```

```
        .then(data => {
          this.waterData = data;
          let waterDataCount = this.waterData.query.count;
      console.log("last temperature recorded: " +
      this.waterData.query.results.row[waterDataCount-1].col1);
          this.waterTemp = this.waterData.query.results.row[waterDataCount-1].col1;
      });
        }
```

10. Nun können wir die Funktion im Constructor aufrufen. Wir wollen auch, dass die App die Funktion aufruft, wenn sie aus dem Hintergrundprozess (Switchen zwischen Apps) kommt. Somit sieht der User immer die aktuellsten Daten. Dazu verwenden wir den EvenetListener resume:

```
    constructor(public navCtrl: NavController, public waterService: WaterData) {
      this.getWaterData();
    //when app is resuemd (switched between apps and come back) then update
data
        document.addEventListener('resume', () => {
          this.getWaterData();
        });
    }
```

11. Jetzt zum optischen! Auf page1.scss füge folgenden Code hinzu falls die App aussehen sollte wie auf dem Intro-Bild, ansonsten kannst du diesen Schritt auslassen.

```
page-page1 {
  .box {
  width: 100%;
  height: 100%;
  box-shadow: 0 2px 30px rgba(black, .2);
  background: #7eddf7;
  position: fixed;
  overflow: hidden;
  transform: translate3d(0, 0, 0);
  bottom:0;
  }


  .wave {
  opacity: .4;
  position: absolute;
  top: 20%;
  left: 50%;
  background: #0af;
  width: 500px;
  height: 100%;
  margin-left: -250px;
  margin-top: -250px;
  transform-origin: 50% 48%;
  border-radius: 43%;
  animation: drift 3000ms infinite linear;
```

```
    }

    .wave.-three {
    animation: drift 5000ms infinite linear;
    }

    .wave.-two {
    animation: drift 7000ms infinite linear;
    opacity: .1;
    background: yellow;
    }

    .box:after {
    content: '';
    display: block;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    background: linear-gradient(to bottom, rgba(#e8a, 1), rgba(#def, 0) 80%,
rgba(white, .5));
    z-index: 11;
    transform: translate3d(0, 0, 0);
    }

    .title-waves {
    position: absolute;
    left: 0;
    top: 120px;
    width: 100%;
    z-index: 1;
    line-height: 75px;
    text-align: center;
    transform: translate3d(0, 0, 0);
    color: white;
    letter-spacing: .1em;
    font-size: 75px;
    text-shadow: 0 1px 0 rgba(black, .1);
    text-indent: .3em;
    }
    @keyframes drift {
    from { transform: rotate(0deg); }
    from { transform: rotate(360deg); }
    }

    .background{
    color: #999999;
    position: absolute;
```

```
    top: 0;
    left: 0;
    z-index: -100;
    }


    .scroll-content {
    margin-top:0 !important;
    }


    .content-ios {
    color: #000;
    background-color: #2a2c31;
    }
    .content-md {
    color: #000;
    background-color: #2a2c31;
    }

    /* make navbar transparent */
    .toolbar-ios-darkergrey .toolbar-background-ios {
    background: rgba(0, 0, 0, 0);
    }
    .toolbar-md-darkergrey .toolbar-background-md {
    background: rgba(0, 0, 0, 0);
    }

    .toolbar-background{
      background:linear-gradient(180deg, rgba(29,29,29,0.4),rgba(0,0,0,0))
  !important;
    }
  }
```

12. Für die View öffne page1.html

```
<ion-header no-border>
  <ion-navbar>
    <button ion-button menuToggle color="light">
      <ion-icon name="menu"></ion-icon>
    </button>
  </ion-navbar>
</ion-header>

<ion-content padding>
  <div class="title-waves">
    <p style="margin:0">{{waterTemp | number:'1.1-1'}}°</p>
    <p style="font-size: 17px;letter-spacing: 0.2em;">Rhy zBasel</p>
  </div>
```

```
</ion-content>
<div class='box' style="position: fixed; bottom:0">
  <div class='wave -one'></div>
  <div class='wave -two'></div>
  <div class='wave -three'></div>
</div>
```

Bei der Expression *{{waterTemp | number:'1.1-1'}}* verwenden wir eine Pipe welcher unsere Temperatur auf eine Stelle nach dem Komma rundet. Diese Pipes sind Teil von Angular aber du kannst auch eigene schreieben (https://angular.io/docs/ts/latest/guide/pipes.html).

13. Done