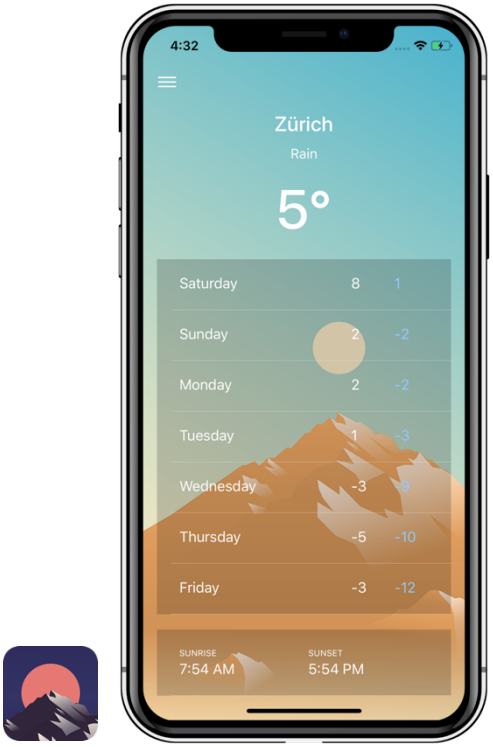
Aufgabe 1

Wir erstellen eine App **Geostorm**, welche Daten durch einen GET-Request erhält. Die Daten werden im JSON Format geliefert und wir bereiten sie auf, damit wir die Temperatur eines Ortes 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. Erstelle eine neue App mit dem Sidemenu Template
    a. `ionic start Geostorm sidemenu`
    b. Tippe „y“ um Cordova in deine App einzubinden
    c. Navigiere in den neu erstellten Ordner
2. Starte die App mit ionic serve
    a. Du solltest jetzt das Standard Sidemenu Template sehen
3. Fügen wir nun einen Provider hinzu um den Rest-Service aufzurufen. Wir können diesen Shared Service auf jeder Page in der App verwenden.
    a. `ionic generate provider RestData`
    b. `unter app.module.ts solltest du jetzt den Provider unter den Imports und in der Providers Array-Liste sehen`

```
1. import { RestDataProvider } from '../providers/rest-data/rest-data';
2. providers: [
    StatusBar,
    SplashScreen,
    {provide: ErrorHandler, useClass: IonicErrorHandler},
    RestDataProvider
  ]
```

4. Wir werden http calls machen, deswegen brauchen wir noch den Import des HttpClientModule

```
a. import { HttpClientModule } from '@angular/common/http';
```

```
b. und beim Imports Array noch hinzufügen
        1.    imports: [
              BrowserModule,
              HttpClientModule,
              IonicModule.forRoot(MyApp),
            ],
```

5. Öffne rest-data.ts und stelle sicher, dass HttpClient importiert ist

```
a. import { HttpClient } from '@angular/common/http';
b. Stelle auch sicher, dass der Constructor die Dependency enthält
        1. constructor(public http: HttpClient) {
c. In Angular nennt sich dies Dependency Injection. Wir laden das Modul
   dort wo wir es brauchen.
```

6. Füge zwei Member Variable zu rest-data.ts hinzu, apiUrl_forecast (für 7 Tage Vorschau) und apiUrl_current (für das Wetter von heute). Setze für die apiUrl_forcast und apiURl_current, die Url zu Wetter Daten von OpenWeatherMap ein (mit deinem Key nach APPID=).

```
a. apiUrl_forecast =
   'http://api.openweathermap.org/data/2.5/forecast/daily?id=7287650&unit
   s=metric&cnt=7&APPID=XXXXXXX';
b. apiUrl_current =
   'http://api.openweathermap.org/data/2.5/weather?id=7287650&units=metri
   c&cnt=7&APPID=XXXXXX';
```

7. Den Typ für die Variable apiUrl_forecast und apiUrl_current kannst du als „any“ setzen (apiUrl_forecast: any = `'http://.... '`. Danach schreibe die Funktion getForecastData() mit dem GET-Call auf die apiUrl.

```
getForecastData() {
  return new Promise(resolve => {
    this.http.get(this.apiUrl_forecast).subscribe(data => {
      resolve(data);
    }, err => {
      console.log(err);
    });
  });
}
```

8. Das gleiche nun um die Daten für das aktuelle Wetter zu holen

```
getCurrent() {
  return new Promise(resolve => {
    this.http.get(this.apiUrl_current).subscribe(data => {
      resolve(data);
    }, err => {
      console.log(err);
    });
  });
}
```

9. Nun kannst du den Provider in jeder Page benutzen, indem du ihn wie folgt in der Page importierst:

```
import { RestDataProvider } from '../../providers/rest-data/rest-data';
```

10. Als nächstes, füge den RestDataProvider deinem Constructor hinzu

```
constructor(public navCtrl: NavController, public restProvider: RestDataProvider) {
```

11. Füge folgende Member Variablen vom Type any oberhabl des Constructors hinzu (mit TypeScript könnten wir hier zwar den genauen Type definieren (string, number, etc.). Einfachheitshalber lassen wir es bei any, bei welchem wir keinen Type prüfen.

```
export class HomePage {

  background_mood: any; //for background image
  city: any; // for city name
  current_weather_all: any; // unfiltered weather data
  current_weather: any; // filtered weather data
  current_hours: any;  // for current ours to calculate when to change bg image
  description: any; // for weather condition description
  sunrise: any; // sunrise time
  sunset: any; // sunset time
  weather_data: any; // forecast unfiltered
  weather_data_list: any; //forecast filtered
```

12. Nun fügen wir drei Funktionen hinzu um die Daten für den Forecast sowie die Daten für den heutigen Tag zu holen(apiURL_forcast, apiURL_current). Wir werden auch die aktuelle Zeit abfragen, damit wir später in unserer App ein Hintergrundbild je nach Tageszeit anzeigen können. Füge dies alles nach dem Constructor hinzu:

```
   getData() {
     this.restProvider.getForecastData()
     .then(data => {
       this.weather_data = data;
       this.city = this.weather_data.city.name;
       this.weather_data_list = this.weather_data.list;
     });
   }

   getCurrent() {
    this.restProvider.getCurrent()
    .then(data => {
      this.current_weather_all = data;
      this.current_weather = this.current_weather_all.main.temp;
      this.description = this.current_weather_all.weather[0].main;
      this.sunrise = this.current_weather_all.sys.sunrise;
      this.sunset = this.current_weather_all.sys.sunset;
    });
  }

  getCurrentHour(){
    var d = new Date();
```

```
    this.current_hours = d.getHours();
    if (this.current_hours >= 8 && this.current_hours <= 18){ //day
      this.background_mood = "Sun.png";
    }else if (this.current_hours >=19 && this.current_hours <22){ //evening
      this.background_mood = "Moon.png";
    }else{ //night
      this.background_mood = "Blood.png";
    }
  }
```

13. Nun können wir die Funktionen 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 restProvider: RestDataProvider)
{
    this.getData();
    this.getCurrent();
    this.getCurrentHour();

    //when app is reopened call this
    document.addEventListener('resume', () => {
      this.getData();
      this.getCurrent();
      this.getCurrentHour();
    });
  }
```

14. Jetzt zum optischen! In home.html erstellen wir das Design der App. Du kannst deine App frei gestalten!
    Anbei ist ein Beispiel, wie die App ausehen könnte. Darin findest du einige interessante Beispiele:

    - Die Expressions mit z.B. {{city}}, womit wir auf unser Objekt city in home.ts zugreifen können. Sobald siech das Objekt in home.ts änder, wird es in der View (home.html) automatisch aktualisiert! The Power and Magic of Angular 😉

    - ngFor="let data of weather_data_list" für eine Schleife durch unser weather_data_list Objekt um die Wettervorhersage anzuzeigen.

    - [ngStyle]="{'background-image':'url(assets/imgs/'+ **background_mood** +')'}" um das Hintergrundbild anzuzeigen, je nach Uhrzeit. Hier verwenden wir statt „style=" [ngStyle], da Angular die Expression **background_mood** auswerten muss. Wenn wir „style=" verwenden, ist zur Laufzeit background_mood noch nicht ausgewertet und wir erhalten einen falschen Wert, nämlich background_mood als Text und nicht als Expression. Die Hintergrundbilder findest du auf dem Git.

    - {{data.dt * 1000 **| date:'EEEE'**}} Hier verwenden wir einen Pipe Filter von Angular,

womit wir einfach aus dem Datum den Wochentag erstellen können. Später werden wir noch einen eigenen Filter schreiben um die Temperatur auf- oder abzurunden.

```
<ion-header no-shadow>
  <ion-navbar transparent>
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title></ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding [ngStyle]="{'background-image':'url(assets/imgs/'+
background_mood +')'}" style="background-repeat: no-repeat; background-size:
cover;">

  <div style="text-align:center;">
    <h2 class="light-color">{{city}}</h2>
    <p class="light-color" style="font-size:16px; margin:0">{{description}}</p>
    <h2 class="light-color" style="font-size:64px; margin-
bottom:20px">{{current_weather}}&#176;</h2>
</div>

<ion-item *ngFor="let data of weather_data_list" style="background-color:
rgba(0,0,0,0.20);" class="light-color">
  <ion-grid>
      <ion-row>
        <ion-col col-8><h2>{{data.dt * 1000 | date:'EEEE'}}</h2></ion-col>
        <ion-col col-2><h2>{{data.temp.max}}</h2></ion-col>
        <ion-col col-2><h2 style="color:#91ccff;">{{data.temp.min}}</h2></ion-col>
      </ion-row>
  </ion-grid>
</ion-item>
<br>
<ion-item style="background-color:rgba(0,0,0,0.20)" class="light-color">
  <ion-grid>
      <ion-row>
        <ion-col col-6><p style="font-size:10px;
color:#fff">SUNRISE</p><h2>{{sunrise | date:'h:mm a':'-0600'}}</h2></ion-col>
        <ion-col col-6><p style="font-size:10px; color:#fff">SUNSET</p><h2>{{sunset
| date:'h:mm a':'+0400'}}</h2></ion-col>
      </ion-row>
  </ion-grid>
</ion-item>
</ion-content>
```

15. In home.scss füge folgenden Code hinzu falls die App aussehen sollte wie auf dem Intro-Bild, ansonsten kannst du diesen Schritt auslassen.

```
page-home {

    .item-inner{
        border-bottom: 0.55px solid rgba(255,255,255,0.2) !important;
    }

    .light-color{
        color:white;
    }

    /* transaprent navbar */
    ion-header {
        .toolbar {
         position: absolute;

          .toolbar-background {
            background: transparent;
          }
          .toolbar-title {
            color: black;
          }
        }
      }

    ion-content .scroll-content  {
        padding-top: $toolbar-ios-height + 40px !important;
        padding-bottom: $toolbar-ios-height !important;
    }

    /* END transaprent navbar */


    .bar-button-default{
        color:white !important;
    }

}
```

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).

16. Nun müssen wir nur noch die Temperatur schön runden. Dazu erstellen wir eine Pipe.
    In der Konsole im root des Projekte gib folgendes ein:
    ionic generate pipe round
    Unter src/pipes/round solltest du jetzt ein round.ts file haben.

Stelle sicher, dass in app.module.ts die neue pipe importiert ist:

```
  import { RoundPipe } from '../pipes/round/round';

  @NgModule({
    declarations: [
      MyApp,
      HomePage,
      ListPage,
      RoundPipe
    ],
```

17. Nun können wir unsere Pipe schreiben damit die Temperatur gerundet wird

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({name: 'roundpipe'})
export class RoundPipe implements PipeTransform {
  transform(value: number, args: string[]): any {
        return Math.round(value);
  }
}
```

18. Wir können sie jetzt in den nötigen Stellen in der home.html Seite einsetzen.

```
<ion-col col-2><h2>{{data.temp.max | roundpipe}}</h2></ion-col>
```

DONE!!!🎉