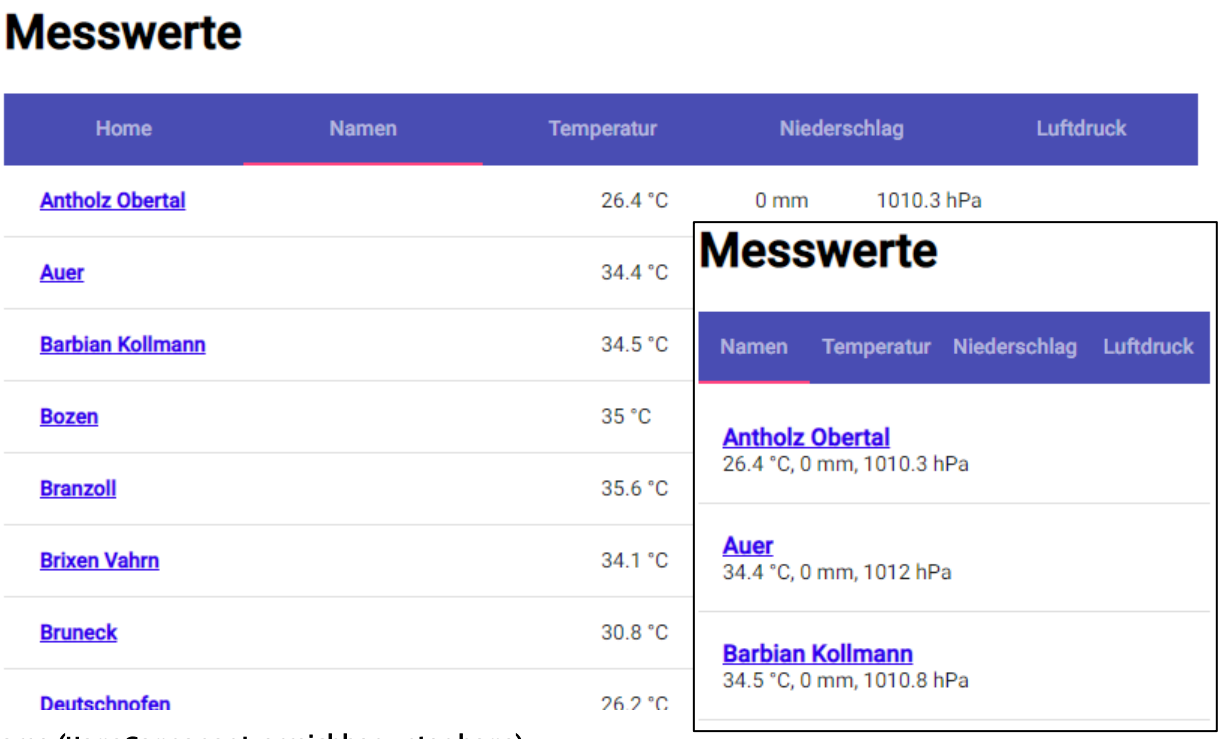
1. Erstellen Sie ein *neues Angular-Projekt*, integrieren Sie das *Routing*, *Angular Material* und das *Flex-Layout*. Binden Sie in dieses den mitgelieferten `WeatherService` ein.

Erstellen Sie dann die im Unterricht vorgestellte Web-Anwendung die folgenden Funktionalitäten aufweisen soll und responsiv die Inhalte darstellen soll:



Home (`HomeComponent` erreichbar unter `home`)

Hier soll ein kurzer erklärender Text zur Webseite angezeigt werden und die Möglichkeit geboten werden, zu den Stationen zu springen. Diese Komponente soll nicht auf sehr kleinen Bildschirmen angezeigt werden.

Stationen (`StationListComponent` erreichbar unter `stations/:sortOrder`)

(**ACHTUNG:** Die Routen sollen hier anders wie im Unterricht besprochen definiert werden). Dabei wird als *Parameter* die *Sortierung* (mögliche Werte: `name`, `temperature`, `precipitation`, `airpressure`) übergeben. Die Komponente holt sich über den `WeatherService` die Stationen in der *gewünschten Sortierreihenfolge* und zeigt diese an. Dazu sollen Sie im `WeatherService` die Methode `getAll()` verwenden. Die Inhalte sollen bei sehr kleinen Bildschirmen über eine `List`- und sonst über eine `Table`-Komponente wie abgebildet dargestellt werden.

Damit die Komponente `StationListComponent` über eine *Änderung des Sortierparameters* informiert wird (diese findet bei der Auswahl eines Menüpunktes statt), soll in der Methode `ngOnInit()` auf die *Parameter der aktuellen Route* (`ActivatedRoute`) ein *Abonnement* (`subscribe`) gelegt werden. Bei einer Änderung der Parameter wird die Komponente darüber informiert. Sie merkt sich in diesem Fall die eingestellte Sortierung und holt sich über `WeatherService` die gewünschte Stationsliste die ebenfalls in der Komponente gespeichert wird, damit sie dann angezeigt werden kann:

```
ngOnInit() {
  this.route.params.subscribe(params => {
    this.ws.getAll(params['sortOrder']).subscribe(stations=>this.stations=stations);
  });
}
```

ACHTUNG: Da es etwas länger dauern kann bis der Web-Service die Stationen in der gewünschten Reihenfolge liefert, der Inhalt des Template aber vorher schon im Browser visualisiert wird, werden Laufzeitfehler auftreten, wenn ohne Kontrolle im Template sofort auf die Stationen zugegriffen wird. Deshalb muss im Template der Fall berücksichtigt werden, wie es sich verhalten soll, wenn die auszugebenden Stationen noch nicht vorliegen.

Durch einen *Klick* auf den Namen der Station sollen deren *Stationsdetails* angezeigt werden.

Stationsdetails (`StationDetailComponent` erreichbar unter `station/:sortOrder/:code`)
Hier sollen die Stationsdetails wie in der Aufgabe des letzten Kapitels angezeigt werden (**HINWEIS:** Beachten Sie, dass für die Route auf Stationsdetails keine Kindroute definiert werden soll, vielmehr werden zwei Parameter übergeben).

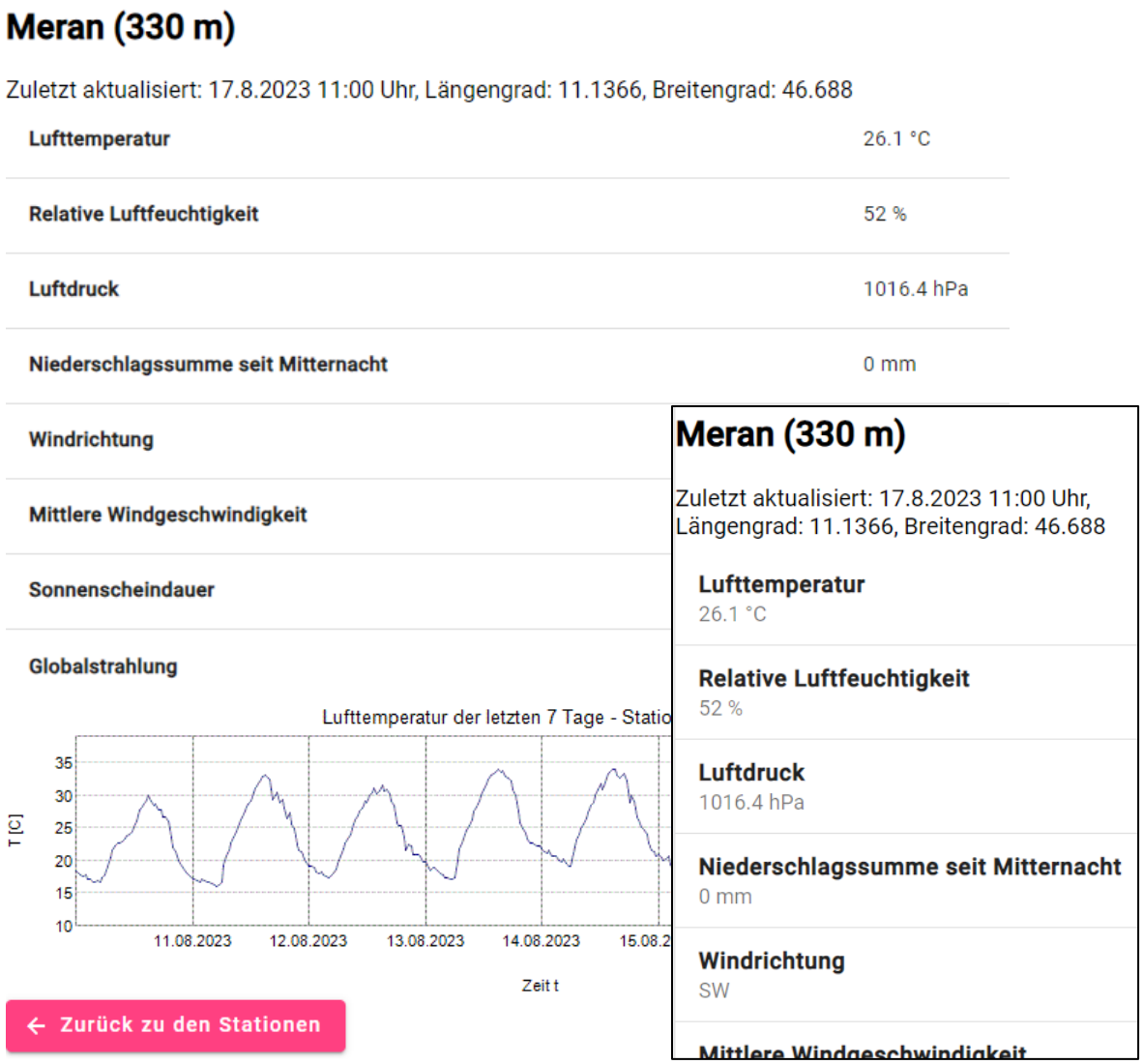
Sie müssen den `WeatherService` um die Methode `get(code: string)` erweitern. Diese Methode holt ähnlich wie `getAll()` nicht alle Stationen, sondern nur eine Station die den übergebenen Stationscode hat (**HINWEIS:** `stations.find(st => st.code === code)` diese Methode liefert die erste Station zurück, welche dem Suchbegriff entspricht).

Die Diagramme in `measurements` sollen *passend zur übergebenen Sortierreihenfolge* angezeigt werden. So soll beispielsweise bei eingestellter Sortierreihenfolge `temperature` nur das *Temperaturdiagramm* angezeigt werden. Werden die Stationen nach Namen sortiert angezeigt, so sollen alle Diagramme angezeigt werden. Auf sehr kleinen Bildschirmen soll kein Diagramm sichtbar werden.

Wiederum sollen die Inhalte bei sehr kleinen Bildschirmen über eine `List`- und sonst über eine `Table`-Komponente angezeigt werden.

In der Komponente soll wiederum anhand eines *Knopfes* zurück in die entsprechende Liste gesprungen werden.

Legen Sie die notwendigen Komponenten an, und realisieren Sie das oben vorgestellte *Routing*. Dabei soll beim Aufruf der Route `stations` ohne Sortierung auf die Liste der *Stationen sortiert nach Name* gesprungen werden.



2. Bauen Sie dann in die `StationListComponent` die im Unterricht vorgestellte `SearchStationComponent` ein. Bei Auswahl einer Station soll in die *Detailansicht gesprungen* werden und diese – in Abhängigkeit der eingestellten Sortierreihenfolge – die Stationsdetails anzeigen.

   In diesem Zusammenhang müssen Sie die Methode `getStations(searchTerm: string)` in `WeatherService` programmieren. Lassen Sie sich dabei von der in der vorigen Aufgabe programmierten Methode `get(code: string)` inspirieren. Beachten Sie die Sortierreihenfolge in der Liste der auswählbaren Wetterstationen.