# Heatmap Technisch Document

In dit document wordt de structuur van de code achter de heatmap doorlopen.

## 1 Frontend

*Home.js*

In dit frontend wordt bij het laden van de pagina middels een *UseEffect* alle heatmapdata opgevraagd uit de backend. Dit wordt uitgevoerd door *axios* calls naar URLs van de API.

* *GetStationsData():* Slaat alle informatie over de meetstations op in *data.*
* *GetAverageData():* Slaat de globale gemiddelde meetwaardes op in *avgData.*
* *GetRegionCords():* Slaat de regio data op in *regionData.*
* *GetHeatMapData():* Slaat de heatmap data op in *TemperatureData* en *FijnstofData.* Dit kan eventueel uitgebreid worden naar meerdere soorten data.

Voor het renderen van de kaart wordt *<Map>* gebruikt van react-leaflet. Deze map heeft een aantal *components:*

* *RegionLayer:* Geeft alle opgehaalde regio’s weer, in verschillende kleuren. Dit zijn *Polygons.* Iedere regio(=polygon) geeft een *Popup* wanneer deze aangeclickt wordt.
* *MeetstationLayer:* Geeft alle meetstations weer. Dit zijn *Markers*. Iedere marker geeft een *Popup* met meer informatie over het meetstation wanneer deze aangeclickt wordt.
* *HeatmapLayer*: Geeft de heatmap weer middels de react-leaflet-heatmap-layer. Meer informatie over hoe dit werkt vind je in de bijbehorende [*documentatie.*](https://www.npmjs.com/package/react-leaflet-heatmap-layer)Er worden nu twee heatmaplayers weergegeven, voor *temperatuur* en *fijnstof.* Dit kan uiteraard uitgebreid worden.

## 2 Backend

### 2.1 HeatmapController

*Ontdekstation013.ClimateChecker/controller/HeatmapController.java*

In de heatmapcontroller wordt door *getDataBySensorType* alle data van één *sensortype* (Temperatuur, Stikstof, Koolstofdioxide, Fijnstof, Luchtvochtigheid, Windsnelheid) verzameld en teruggegeven aan de frontend. Hiervoor moet het *typeid* van het gewenste *sensortype* worden meegegeven.

Deze functie wordt door de frontend nu tweemaal aangeroepen, voor temperatuur (typeid 1) en fijnstof (typeid 4).

### 2.1.1 Services

De HeatmapController maakt gebruik van de SensorService, StationService en LocationService. Er wordt slechts gebruik gemaakt van eenvoudige get/find calls wat verder geen uitleg nodig heeft.

### 2.2 RegionController

*Ontdekstation013.ClimateChecker/controller/RegionController.java*

In de regioncontroller wordt door *GetAll* alle regios(alleen naam en id) teruggegeven aan de frontend. Deze wordt momenteel niet gebruikt.

Door *getAllregionCords* wordt alle informatie over de regios, inclusief coördinatendata en gemiddelde waardes, teruggegeven aan de frontend. Dit wordt afgehandeld door de *regionCordsService.*

### 2.2.1 RegionCordsService

*Ontdekstation013.ClimateChecker/services/RegionCordsService.java*

In de *RegionCordsService* wordt alle relevante regio-data verzameld in *RegionInfoDtos* door de *GetAll.*

De RegionInfoDto bevat:

* *Region:* Dit is de naam en het ID van de regio.
* List<RegionAverageDto>: Lijst met de gemiddelde waardes van alle meetstations in de regio.
* List<double[]> cordsList: Lijst met alle coördinaten van de regio. Deze worden gebruikt om te bepalen welke meetstations in de regio vallen (backend) en om de regio weer te geven in de leaflet (frontend).

De functie *getStationsInRegion* bepaald aan de hand van de coördinaten van de regio welke stations er allemaal binnen de regio vallen. De lijst met stations die wordt teruggegeven wordt gebruikt om de gemiddelde waardes van de regio te bepalen in *getAverageRegionData.*