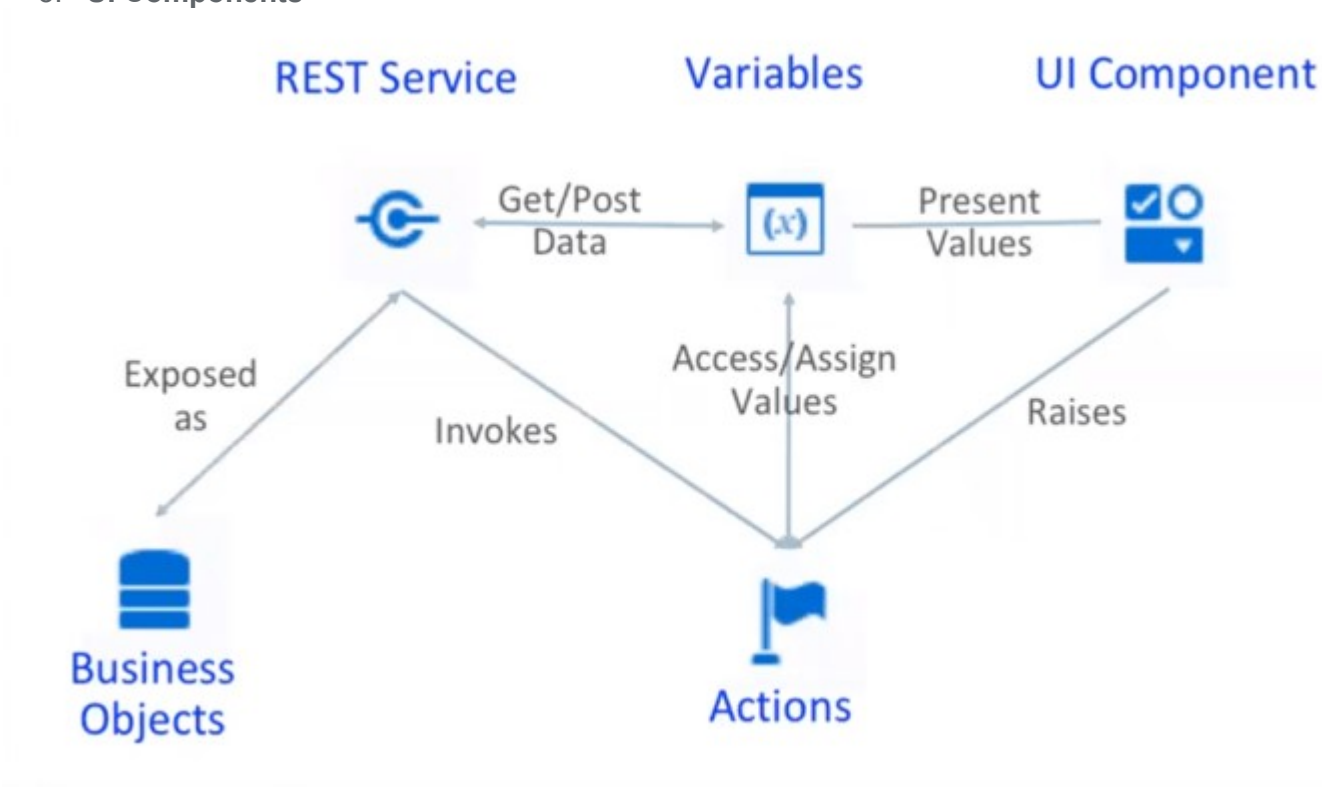
In may 2018 Oracle introduced the new version of Visual Builder Cloud Service. This version is not just aimed at the Citizen Developer, in the end an experienced JavaScript can do nice things with it.

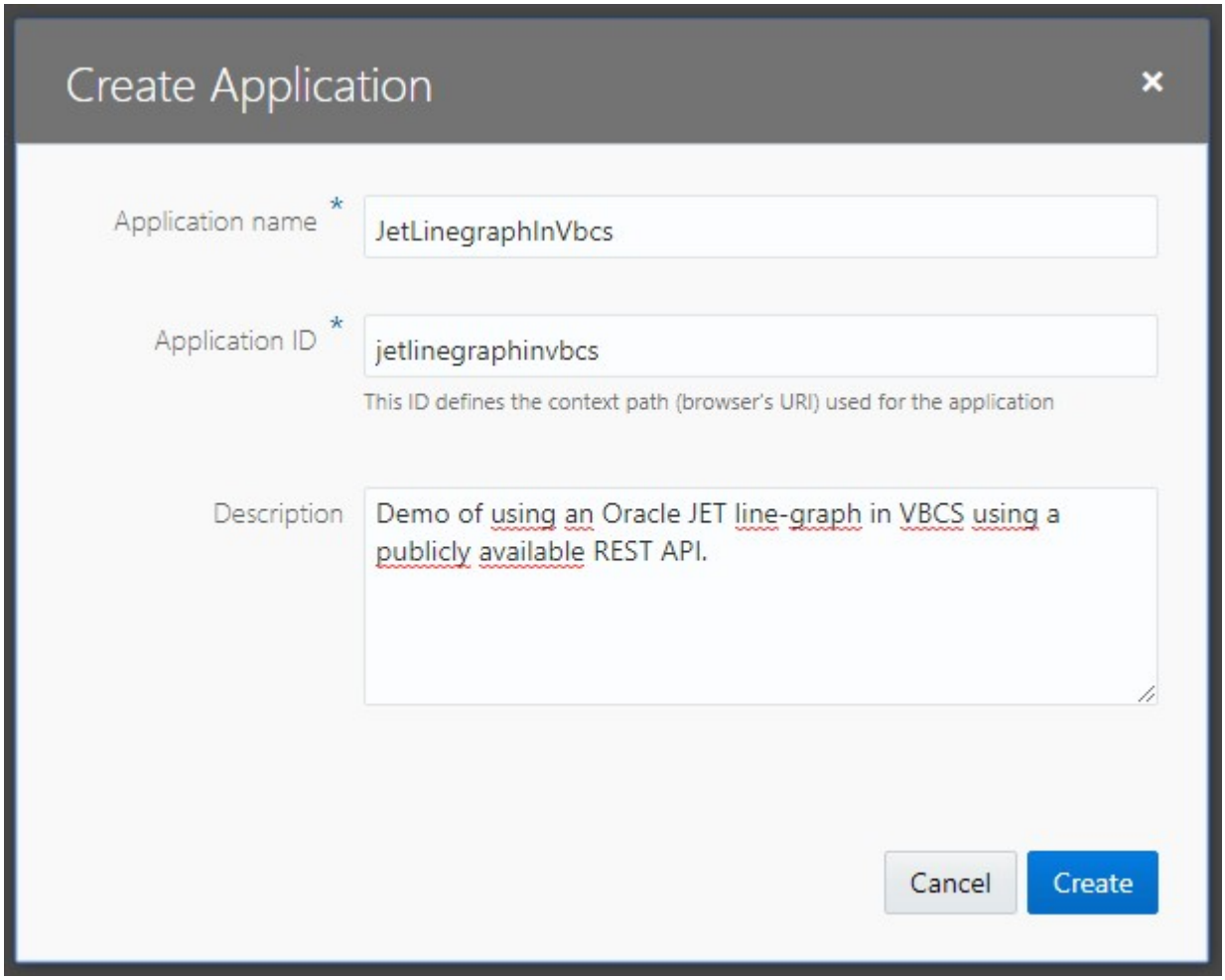
In this blog I will have a look at 5 of the 6 main building blocks you build a VBCS applications with:

1. **REST service connections**
2. **Flows and Pages**
3. **Variables**
4. **Action Chains**
5. **UI Components**



Putting all of this in one blog is a lot, so this is a lengthy one. The final result can be found here.
With VBCS you can create a lot using Drag and Drop! But in the end you have to be aware that it is all Javascipt, HTML5 and CSS you are creating. And it is all build on JET!

Before we can start with these concepts, I create a New Application.

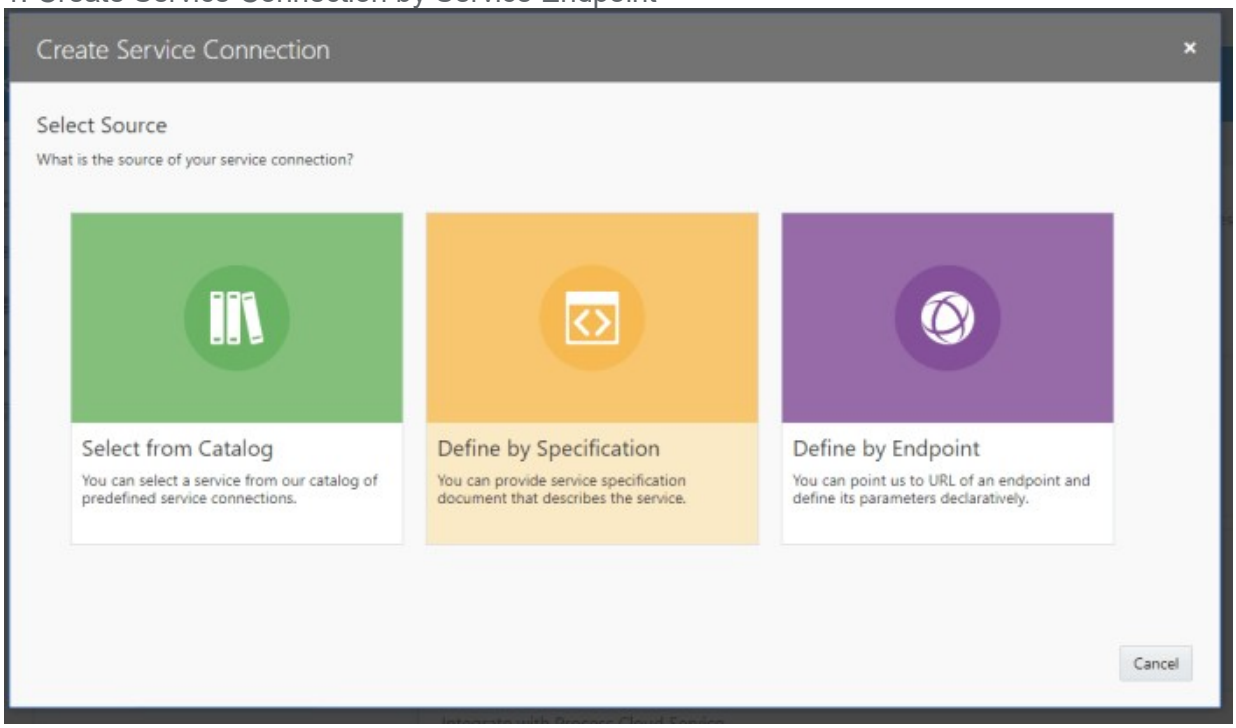


Rest Service Connection

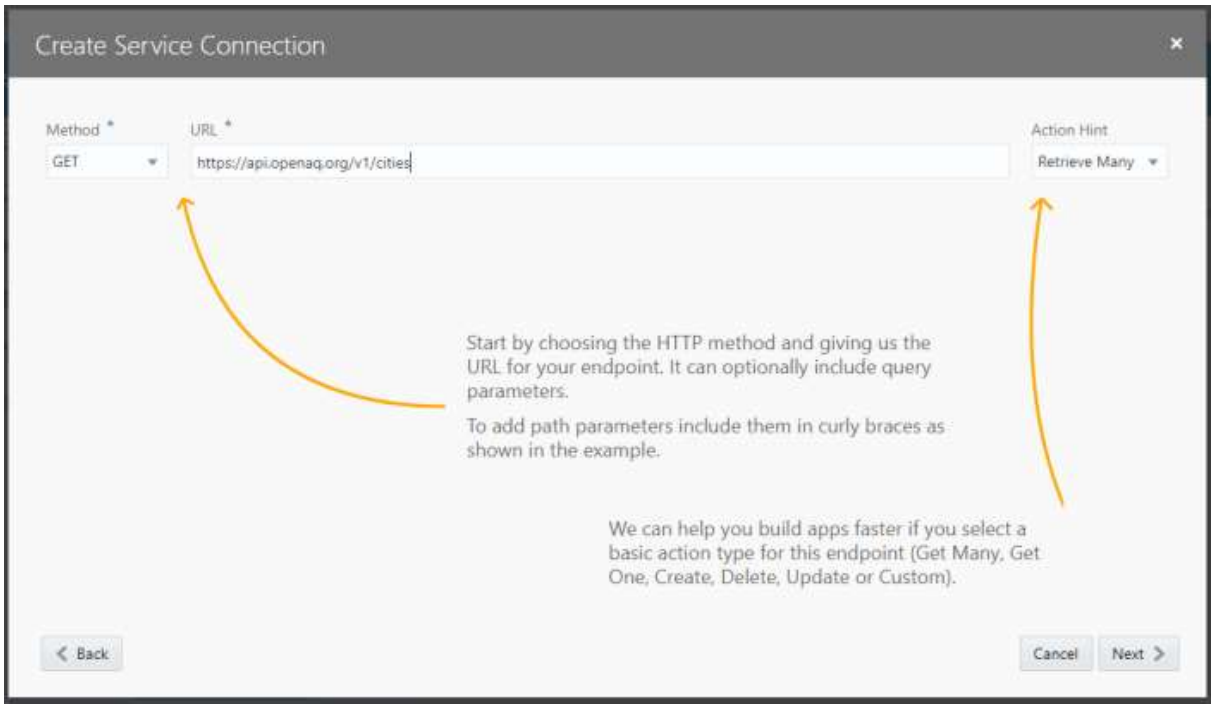
I start with creating some endpoints for a publicly available REST API, https://docs.openaq.org/ An API with Air quality measurements
This API contains several endpoints, the graph I am going to create uses the measurements API. As I am from the Netherlands, I use data from the Netherlands in this blog.

First I create the Service Connection based on the Cities endpoint.

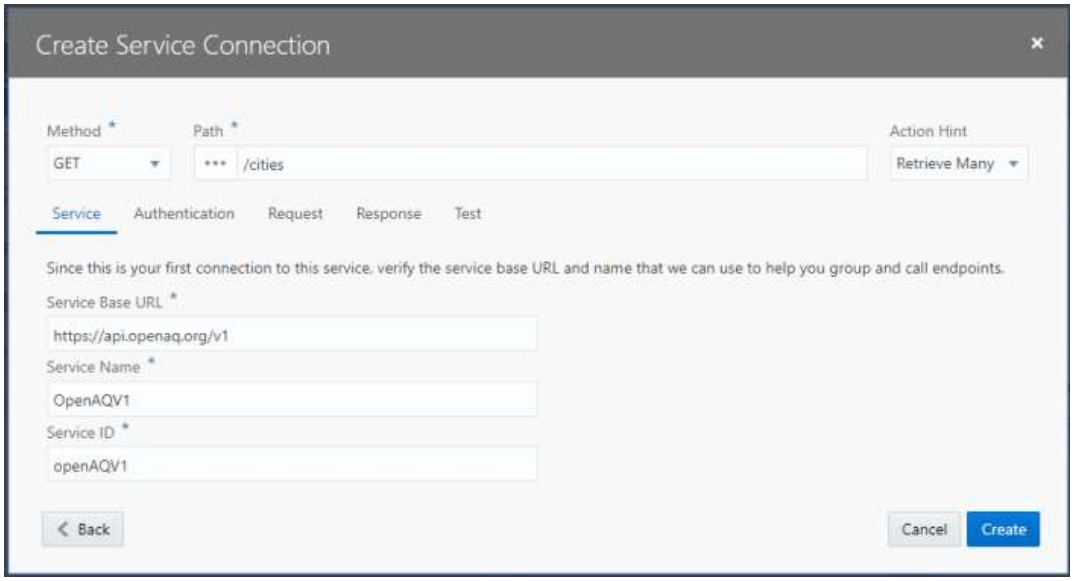
1. Create Service Connection by Service Endpoint



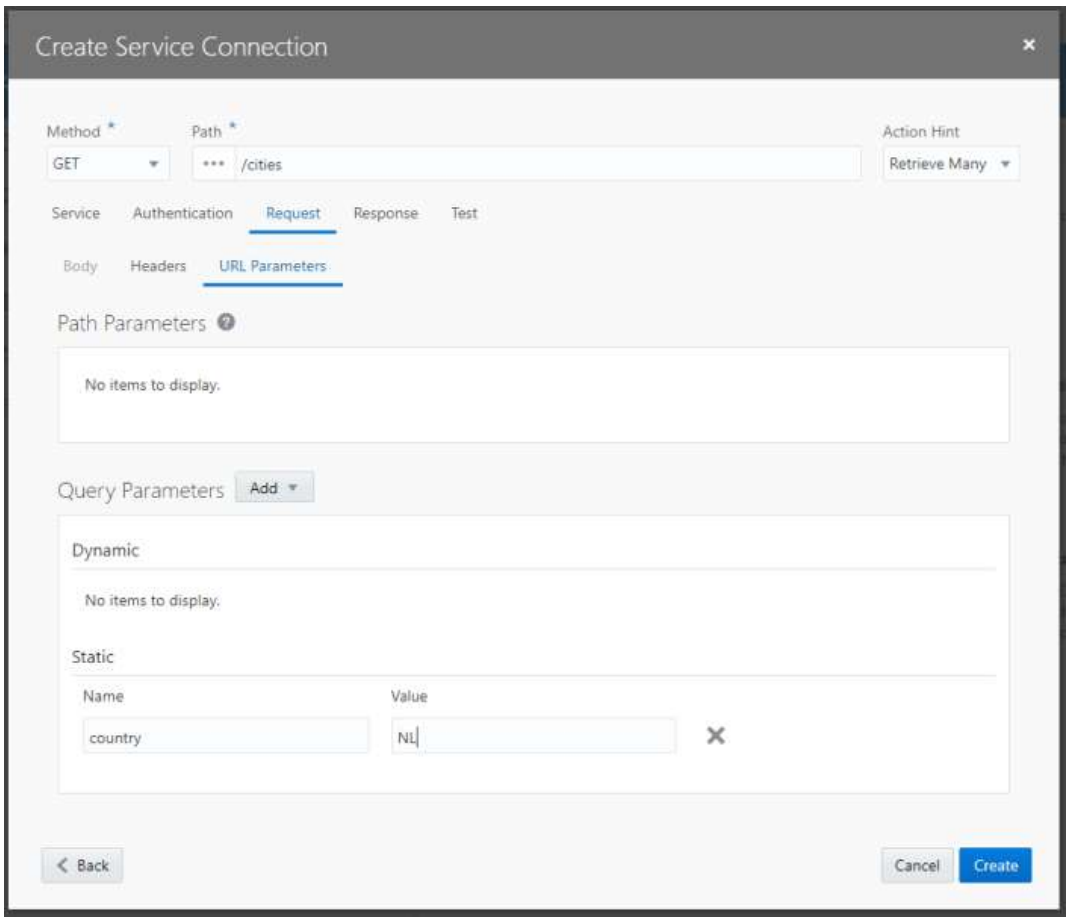
2. Specify “URL”: https://api.openaq.org/v1/cities



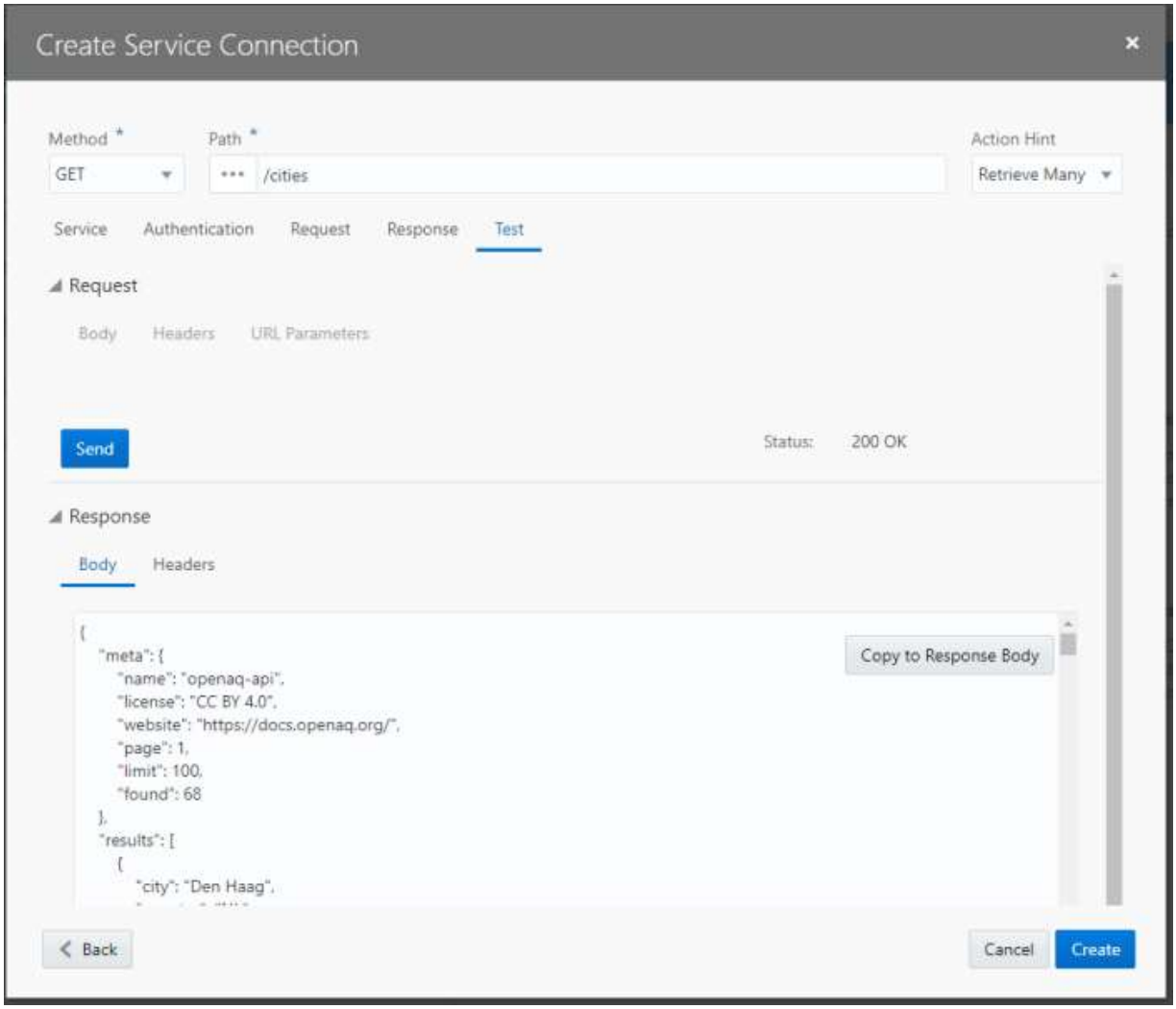
3. Service Base URL is populated with https://api.openaq.org/v1/, give Service proper name



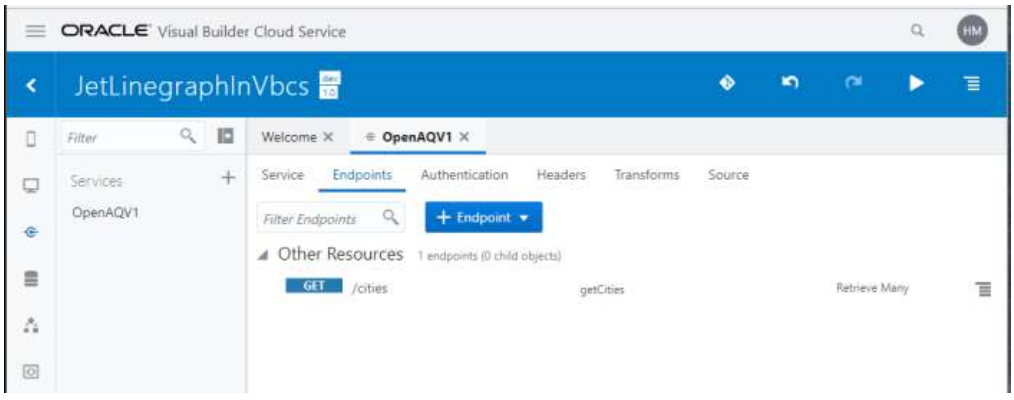
4. For Request > URL Parameters Add the Static Parameter “country” with value “NL”



5. Test the endpoint and Copy to Response Body

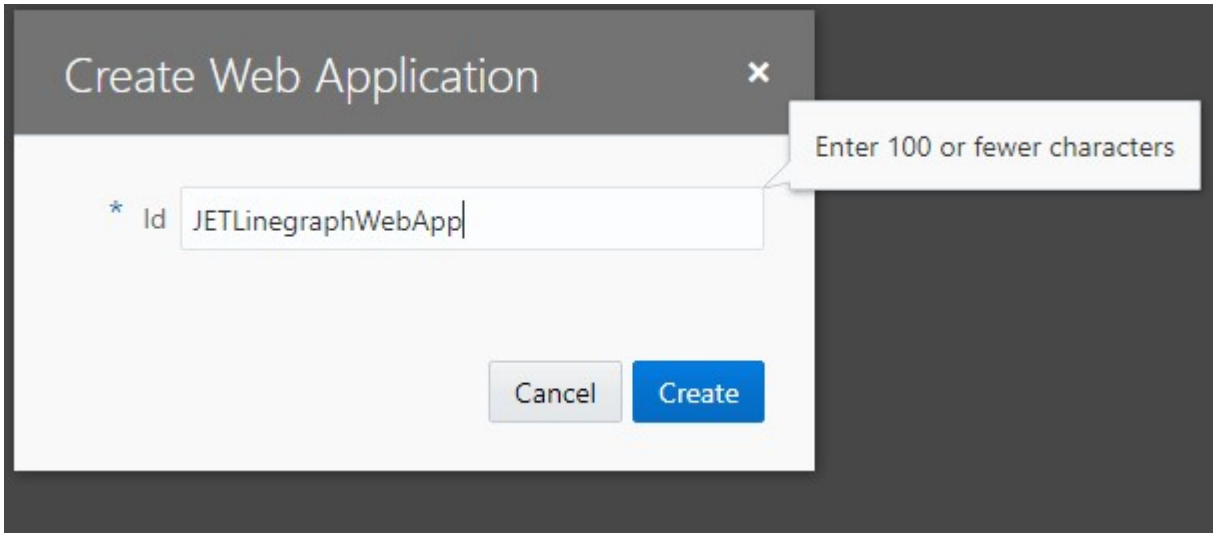


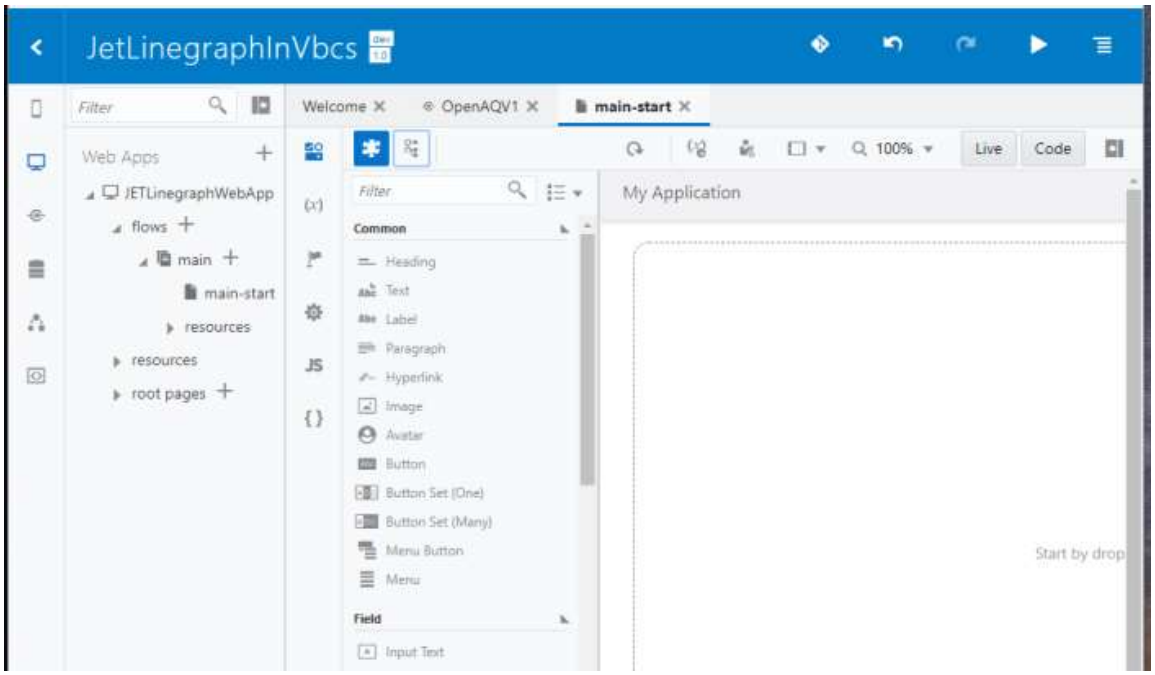
6. Create the endpoint



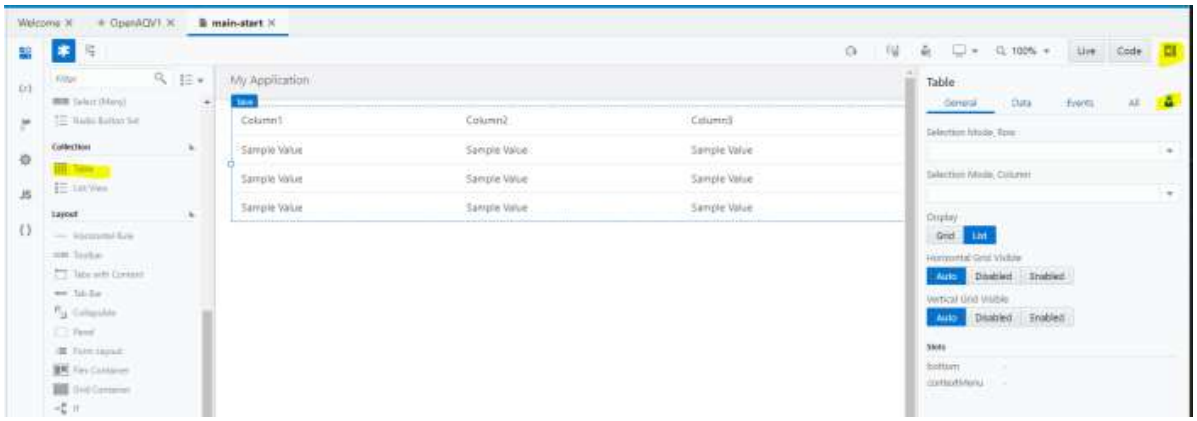
Create flow and first Page

I will create a Web Application that contains the main Flow and main-start Page.



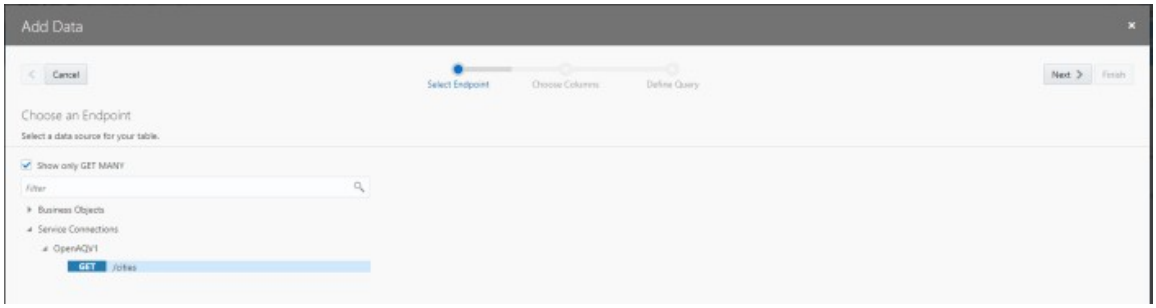


On the main-start Page I drop a Table UI Component.

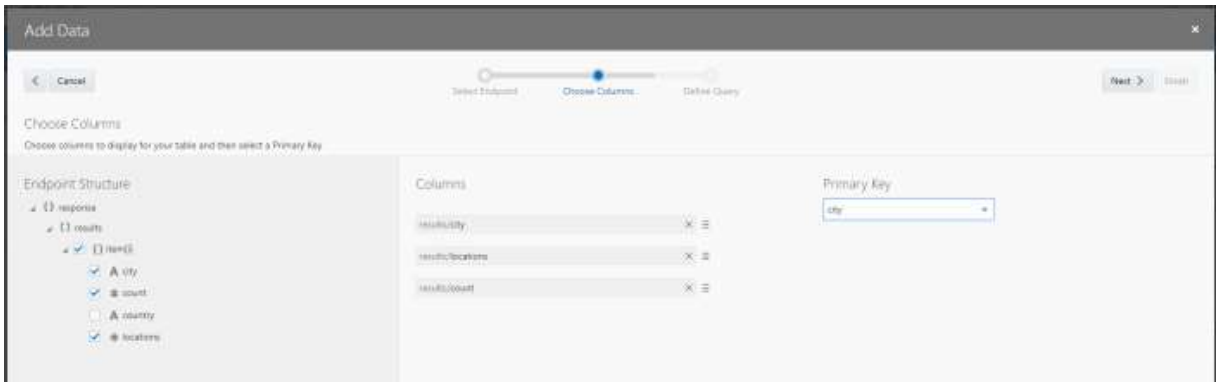


I have marked yellow the Table UI component, the Collapse/Expand Property Inspector and Quick start button.

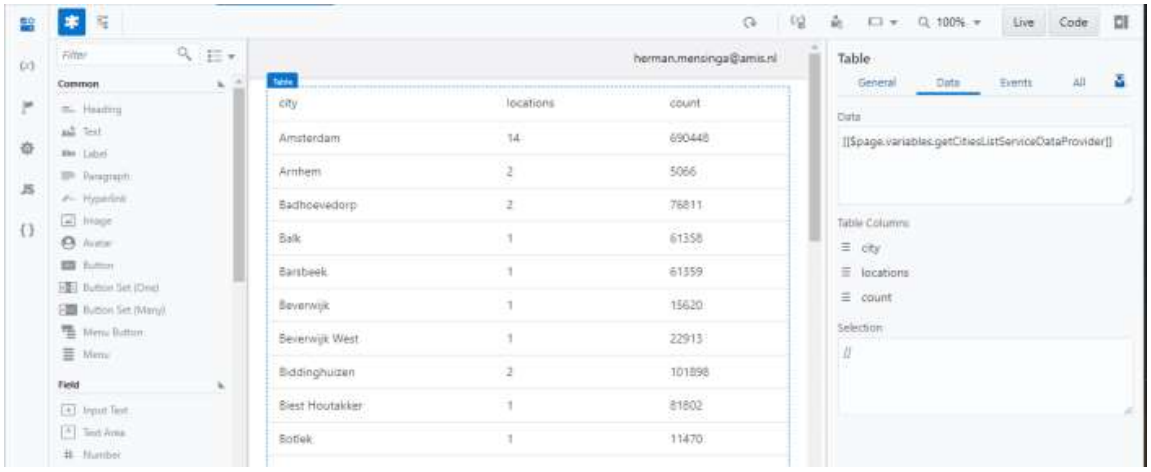
For this Table we Add Data, which is available on the Quick start Tab. Select the cities Service Endpoint.



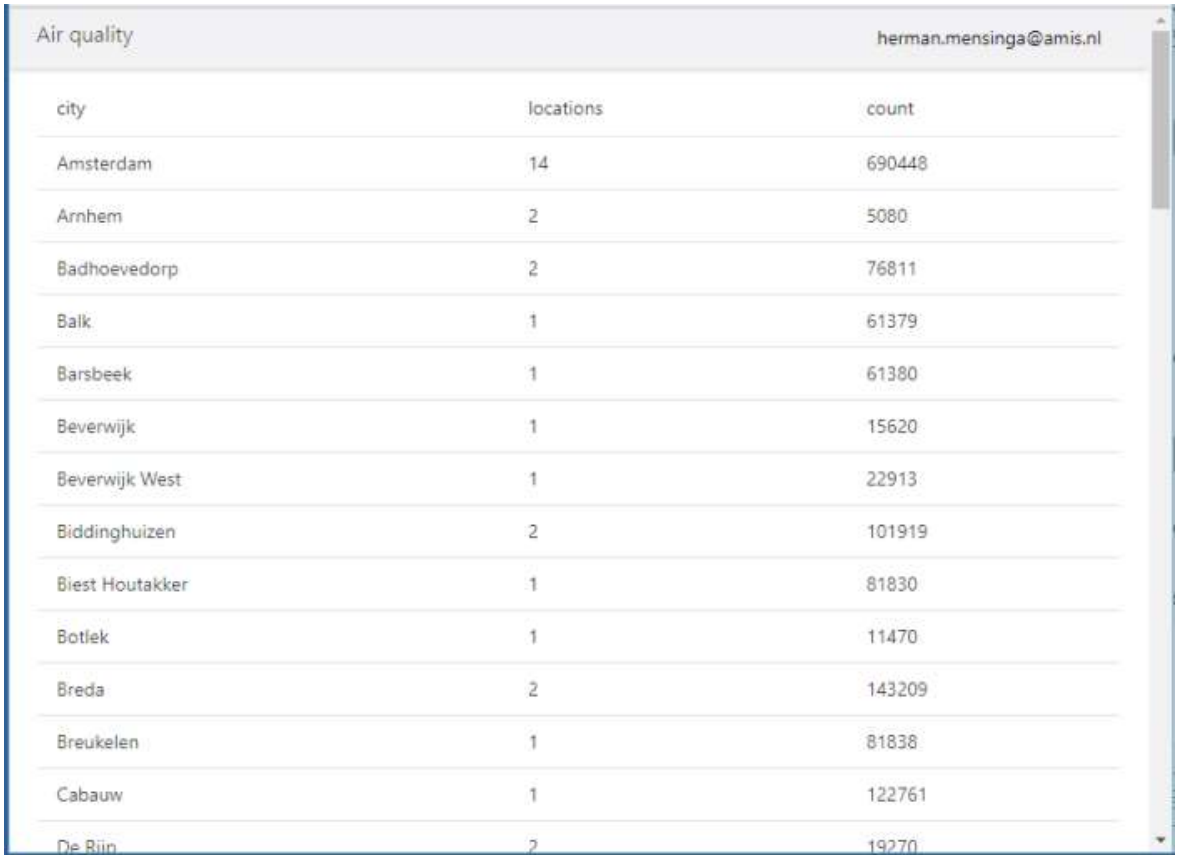
As the country is hardcoded, I won't display it in the table. I reorder the columns with drag and drop. City I select as Primary Key.



In the layout editor the Data from the Service endpoint is displayed. In the code you will see that an Oracle JET oj-table component is used.

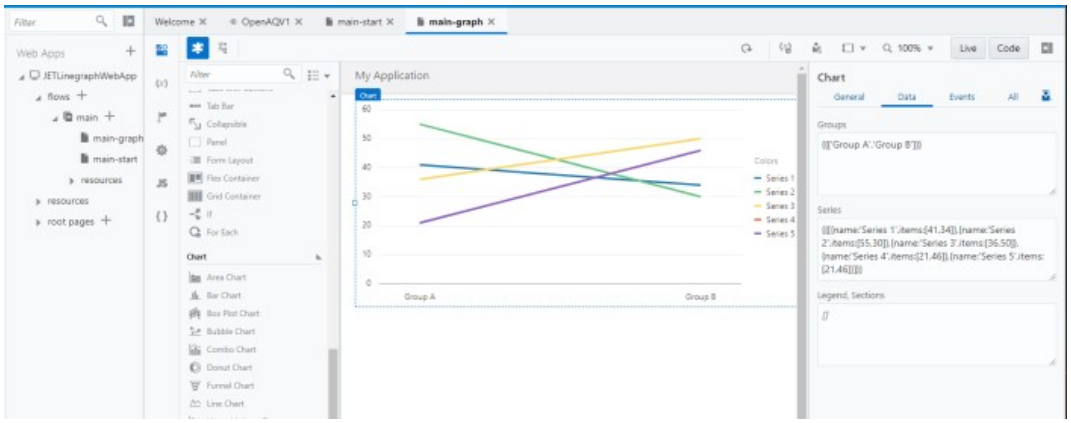


You can also run the Application:



city	locations	count
Amsterdam	14	690448
Arnhem	2	5080
Badhoevedorp	2	76811
Balk	1	61379
Barsbeek	1	61380
Beverwijk	1	15620
Beverwijk West	1	22913
Biddinghuizen	2	101919
Biest Houtakker	1	81830
Botlek	1	11470
Breda	2	143209
Breukelen	1	81838
Cabauw	1	122761
De Bilt	2	19270

Next we add a Page for the Line-graph and drag an Oracle JET Line Chart on it.



Variables and Types

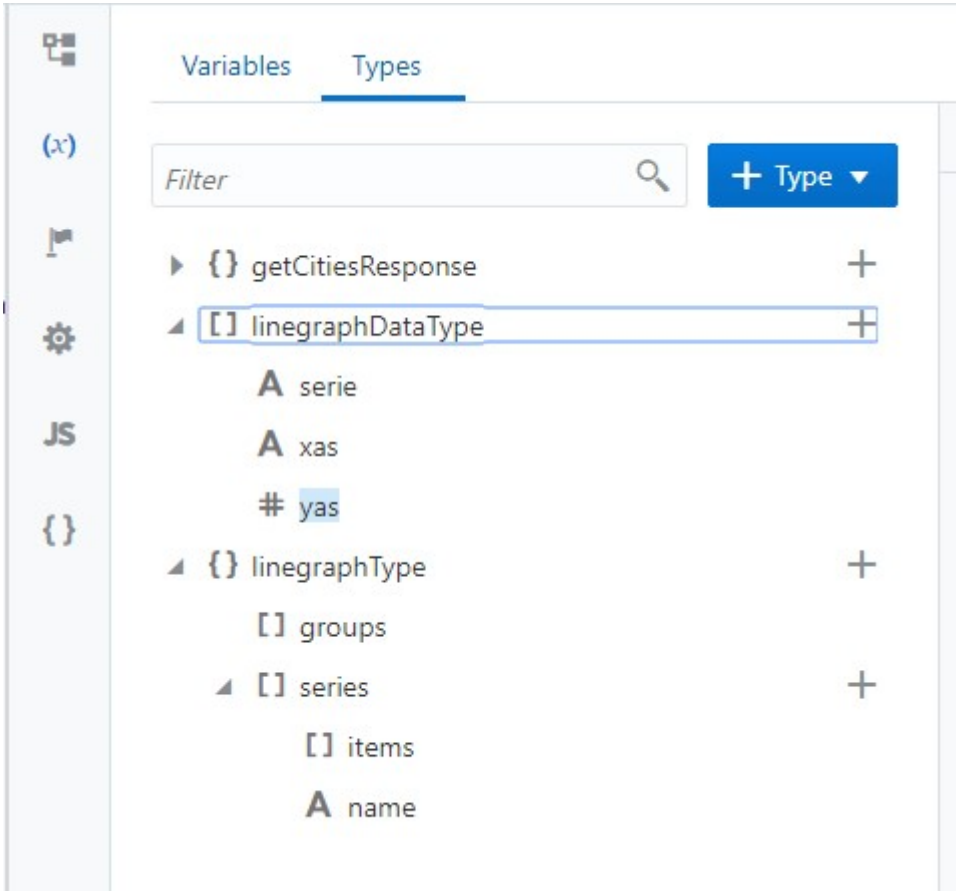
The responses from a Rest endpoints are stored in Variables, UI components and Action Chains use Variables.

When you have a look at the Line Chart code, it contains two Arrays: Groups and Series. The Groups Array is an Array of Strings ([‘Group A’,’Group B’]), the Series array is an Array of Objects ([{name:’Series 1′,items:[41,34]},{name:’Series 2′,items:[55,30]},…]). The Serie Object consists of a String (name) and a numeric Array (items).

For the line Graph I create two Types in the main Flow.

1. a Type with a structure that can hold the result data of a REST call
2. a Type with a structure that can be mapped to a JET line graph

A getCitiesResponse type was already created by VBCS for the response of the REST call. This is the final result I want:



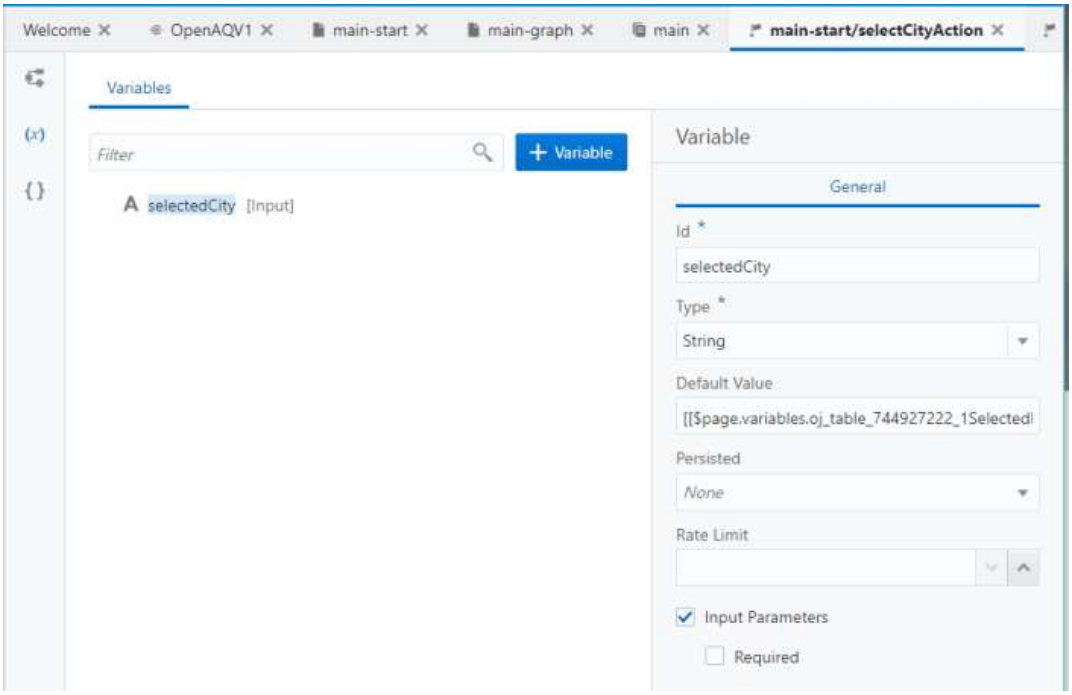
Action Chain

I create an Action chain that will do these steps:

- Show notification with City selected
- Call measurement REST endpoint and map to linegraphDatatype
- Map linegraphDatatype to linegraphType using JavaScript
- Navigate to page line-graph page

When I open the Actions Tab for main-start Page, I see that a Action-Chain was already created. This Action-Chain saves the Primary Key of the row selected in my city Table.

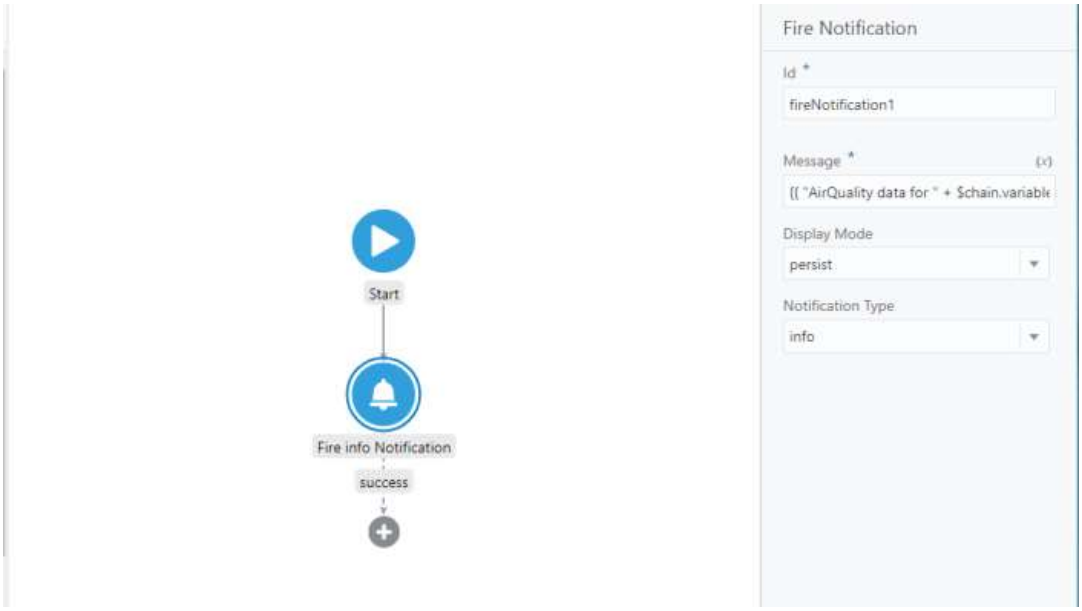
I now create the mentioned Action Chain. In this ActionChain I create a variable and assign the page variable with the selected City as Default.



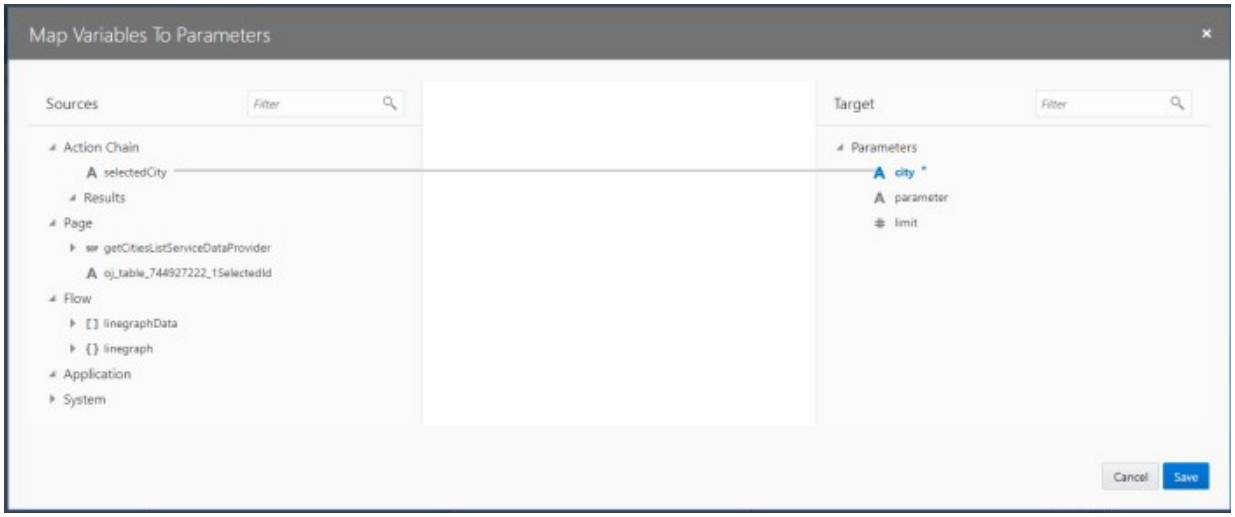
Next I drop a Fire Notification Action on the +-sign below Start.

I set the Display Mode to transinet and specify the Message as

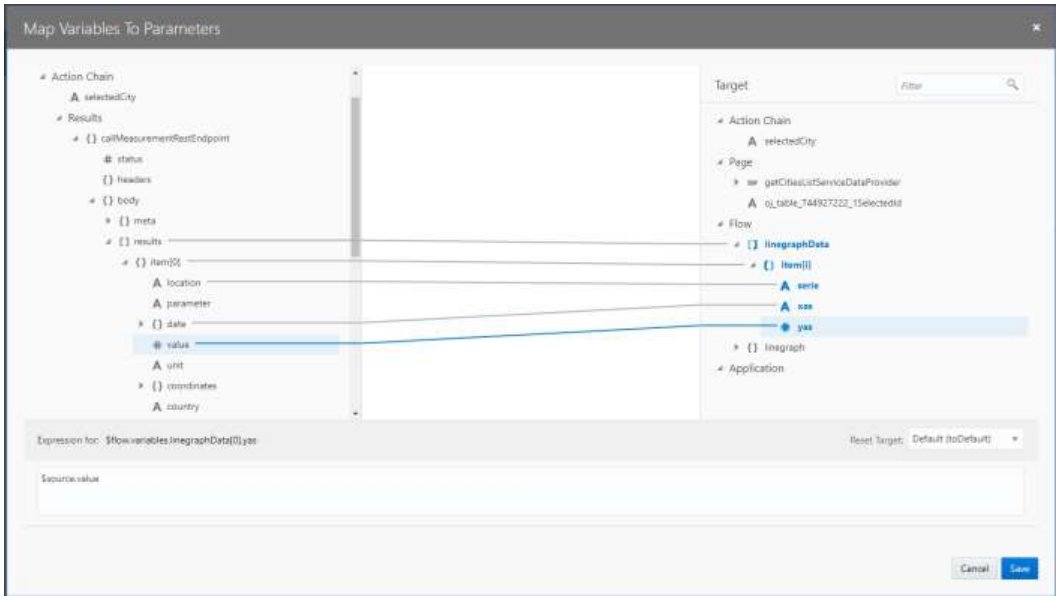
{{ “AirQuality data for ” + $chain.variables.selectedCity + ” is being retrieved.” }}



The measurement REST endpoint is called with a Call Rest Endpoint Action. selectedCity from the Action Chain is mapped to the city parameter of this Action.



The Result of this Action has to be mapped to linegraphData variable using Asign Variables Action.

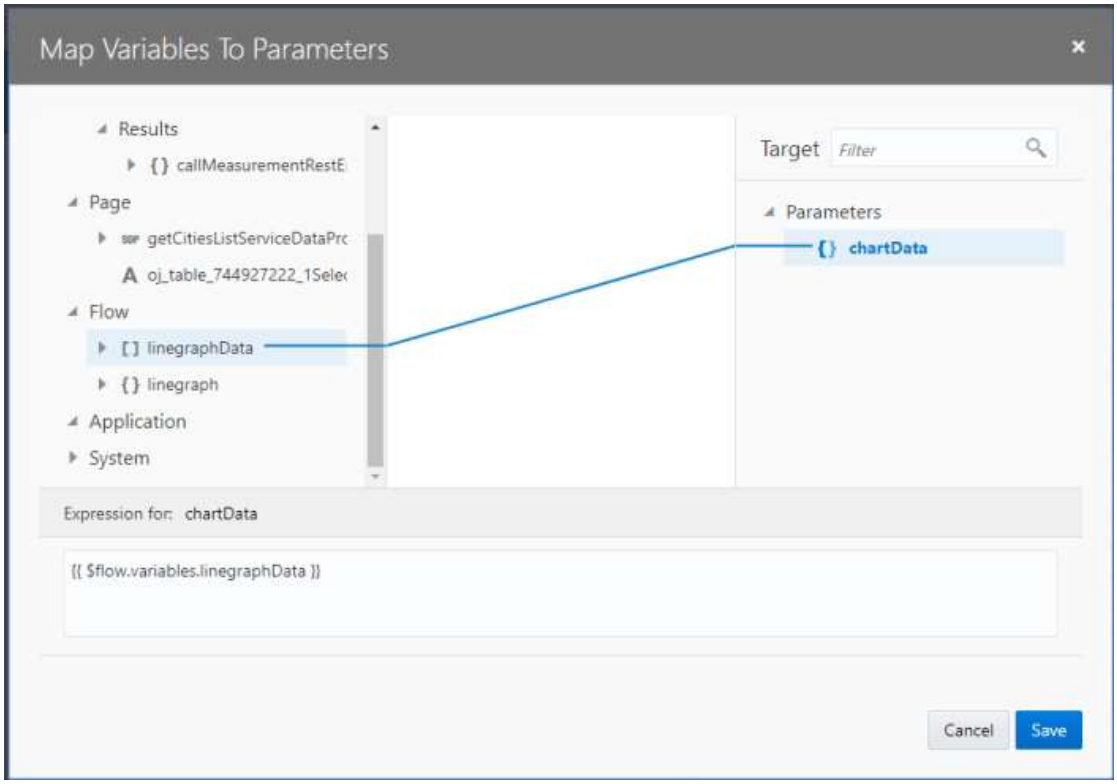


This linegraphData array I need to convert to my linegraph object. For this I will call a peace of javascript. First I create a function in the main Flow javascript.

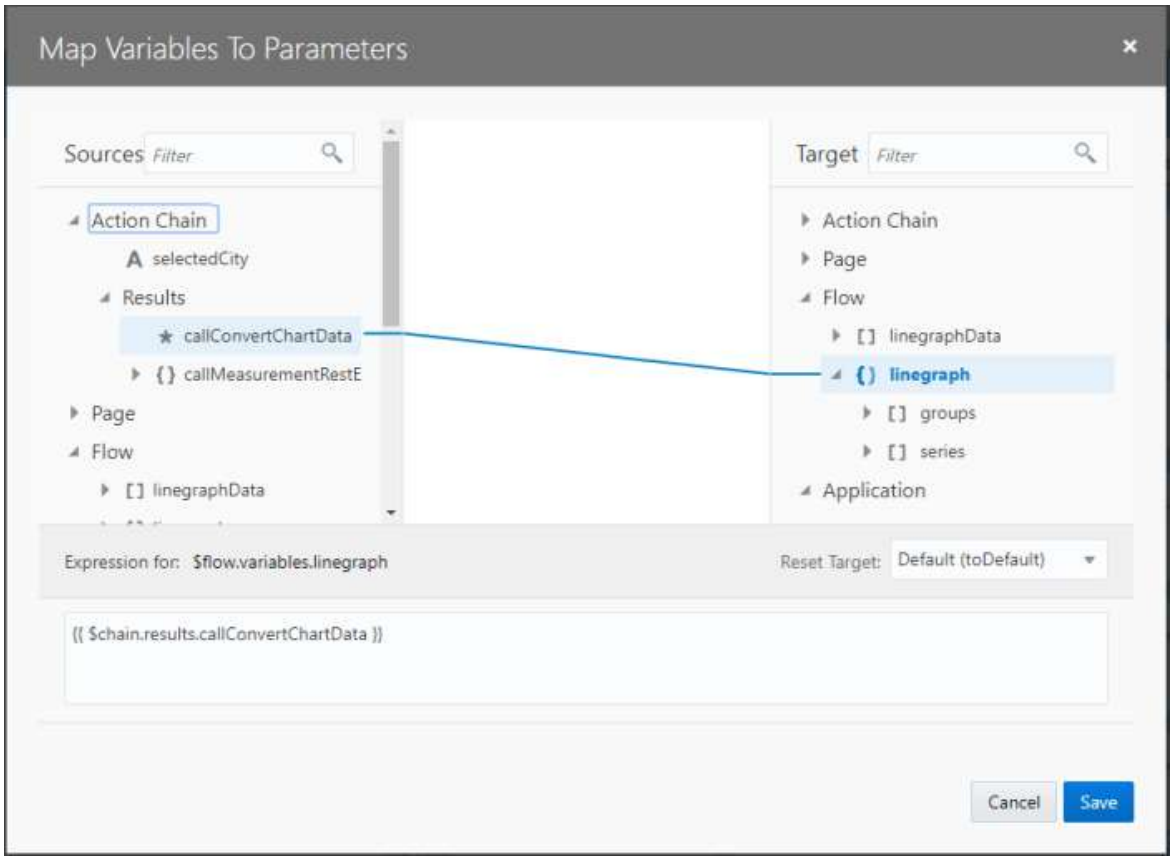
```
define([], function() {
  'use strict';

  var FlowModule = function FlowModule() {};

  FlowModule.prototype.convertChartData = function(chartData) {
    var self = this;
    var startArray = chartData;

    var xas = startArray.map(function(obj) {
      return obj.xas;
    });
    var uniqueXas = xas.filter(function(x, index) {
      return xas.indexOf(x) === index;
    });

    var startSeries = startArray.map(function(obj) {
      return obj.serie;
    });
    var uniqueSeries = startSeries.filter(function(serie, index) {
      return startSeries.indexOf(serie) === index;
    });

    var series = [];
    uniqueSeries.forEach(function(serie) {
      var items = [];
      uniqueXas.forEach(function(x) {
        var obj = startArray.find(function(item) {
          return (item.serie === serie && item.xas ===
            x);
        });
```

The complete peace of javascript can be found in the Application Export that is attached to this blog.

This javascript function can be called with a Call Module Function Action, VBCS recognizes the function added in the javascript. The linegraphData variable needs to be mapped to the chartData parameter.



The result from the javascript function needs to be mapped to the linegraph variable using an Assign Variables Action.

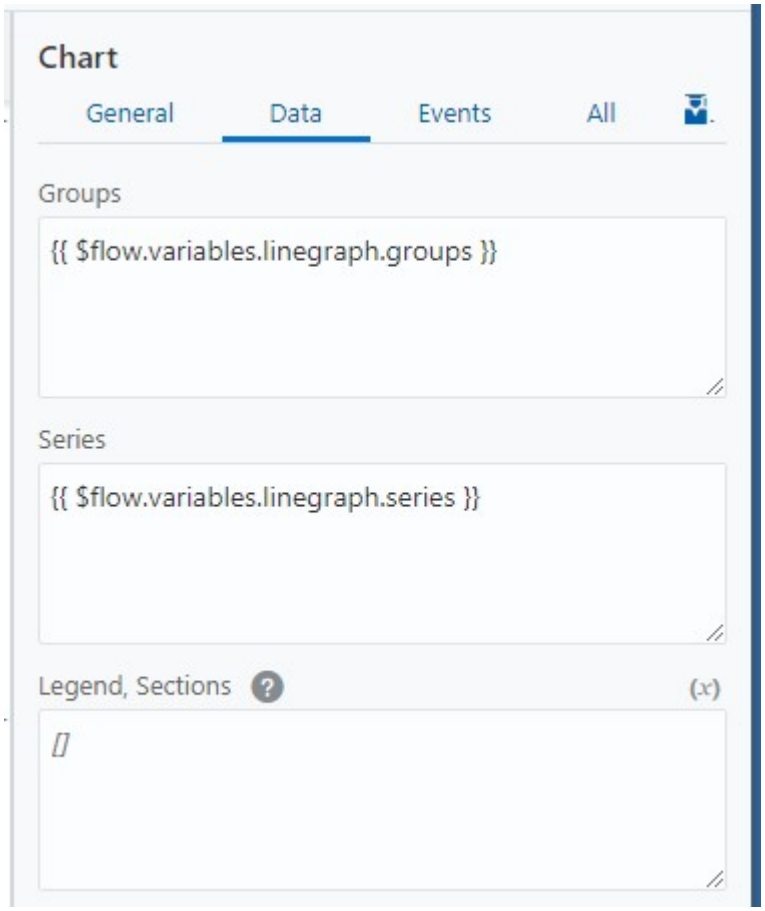


Finally I navigate to the main-graph Page using a Navigate Action.

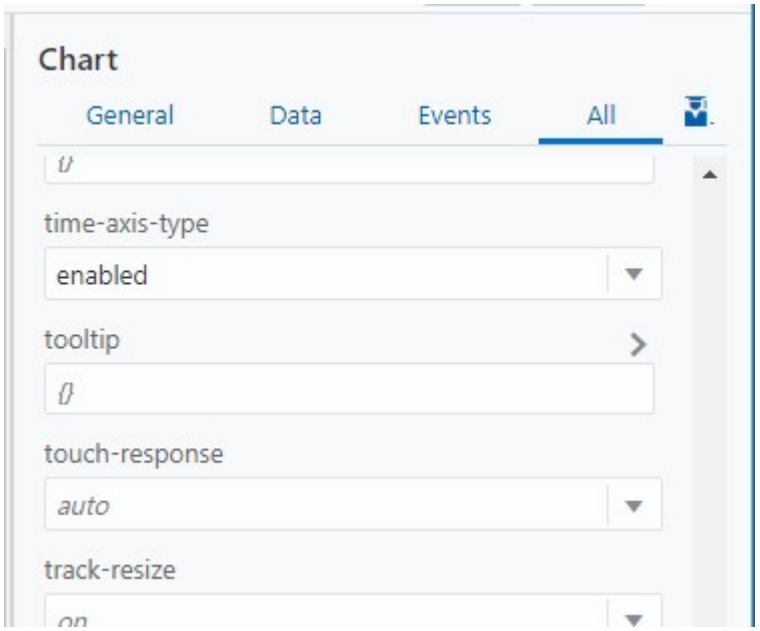
A quick way to call this Action Chain (and get the linegraph) is by calling it from the already existing Action Chain to handle the selection of a row in the Cities table. I add a Call Action Chain Action

UI Components

The linegraph variable is now ready to be used by our graph. In the Data Tab of the Chart we set the Groups and Series.

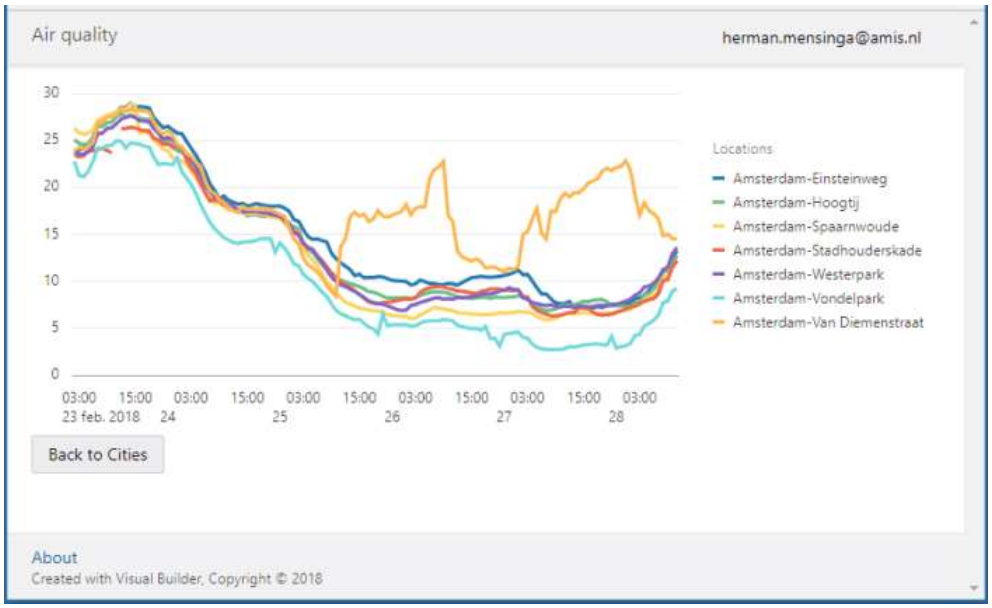


To get a readable layout for the date-axis, I enable the x-axis as time-axis-type:



Everything together

The final graph for Amsterdam:



Amsterdam pm25 graph

The VBCS export can be downloaded here