Deloitte Technical Challenge
GlobalWeather API

Initial Considerations:
The Given GlobalWeather Web service has 2 operations we will need to implement, GetWeather and GetCitiesByCountry each with their own input parameters that need to be handled.

This web service will need to be transformed into a suitable format for a REST API to consume, i.e. JSON from the SOAP response.

Then the REST API will need to be implemented using RAML on the two operations. This would be:

1. A GET request for the weather, having two parameters (countryName, cityName)
   Path: /api/weather/{countryName}/{cityName}
2. A GET request for cities, having a parameter of countryName
   Path: /api/cities/{countryName}

Solution:
My goal was to split learning the task into:

1. Consuming a SOAP web service
2. Developing the REST API using RAML and generating flows

My method for the task is as below:

Method:

1. Create an RAML file defining the API from the operations derived from the WSDL
2. Generate mule flows via that RAML.
3. For each method, set the required input parameters for the WSC operations (GetWeather, GetCitiesByCountry)
4. Get the SOAP XML response and transform the message into the required JSON format

NOTE: I have left the response of the GET request for /weather/{countryName}/{cityName} as the working example, working mainly on converting the /cities response.

Challenges:
Having used API's constantly in my current role, I haven't developed my own API from the ground up, nor been exposed to Mulesoft/Dataweave expression language. This has been a very exciting challenge that, regardless of the outcome, has been a great learning experience.

I decided to learn this in 2 parts, firstly, developing a simple RESTful API using RAML on a sample resource, secondly, consuming a SOAP web service using Web service consumer and transforming/logging the message. Then, combining the two as needed in the given task.

Transforming the XML response was the most challenging, as I couldn't determine how to use DataWeave to transform the response to JSON. I assume there is a typing issue causing this, either in a transform message component or the Web service Consumer requires a schema. However, I believe once this transformed into JSON, the API would respond correctly, as required.

Extensions:

1. More detailed error handling within the RAML and the generated flows.
2. Basic authentication for the GlobalWeather RESTful API. (i.e. API key)
3. Pagination – Limiting the number of objects returned by the request
4. Adding an ID to the JSON response as a unique identifier