Weather Application Report

Advanced JavaScript Continuous Assignment 1

George Blanaru

N00153498

Contents

Introduction

The purpose of this project is to create a working web application that fetches data from an API using the React library. The application must be modular and should be able to handle data of all sizes. To demonstrate that, this project uses Open Weather API combined with Unsplash to create a interactive weather application.

Implementation

The application has several components that handle the functionality. These components are loaded when the user interacts with the application. Components like *App* and *NavigationBar* are loaded from the start while others are loaded when the application requires them, e.g. The Graph is shown when the user inputs a valid city. Appendix A below shows the wireframes created for this application which offer a more in depth look of how the application is generated.

Components

App.js

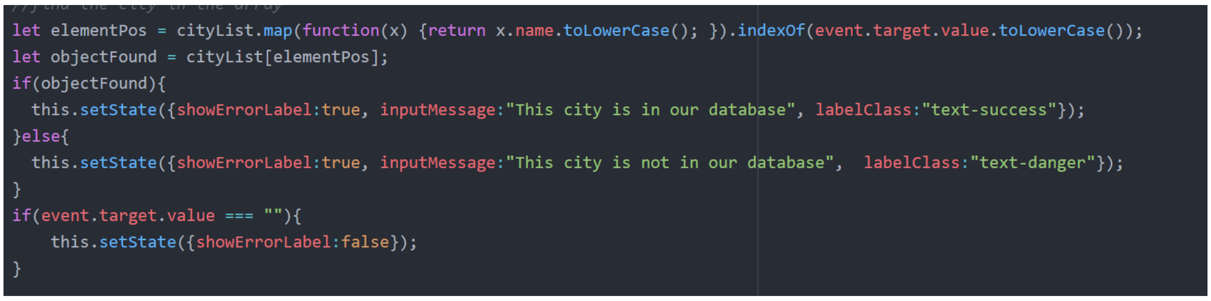
The App component handles the body of the application and loads all the CSS rules. When the client first opens the application, the App component is loading the <NavBar /> and <Home /> and applies styles to the page. The <BrowserRouter /> and the <Route /> components are also present in the <App/> to facilitate the use of links and switching between pages. These components, along with the <Link /> from the <NavBar /> are predefined and loaded from a library called “react-router-dom”.

NavBar.js

The only job of this component is to handle the navigation links. When clicking one of the links, the component is loaded underneath. The reason the component has a state is to add CSS styling to let the user know which link is clicked. An existing and known bug is if the user directly accesses one of the links, the CSS styling is not loaded, as the state is not being set.

CurrentWeather.js and WeatherResult.js

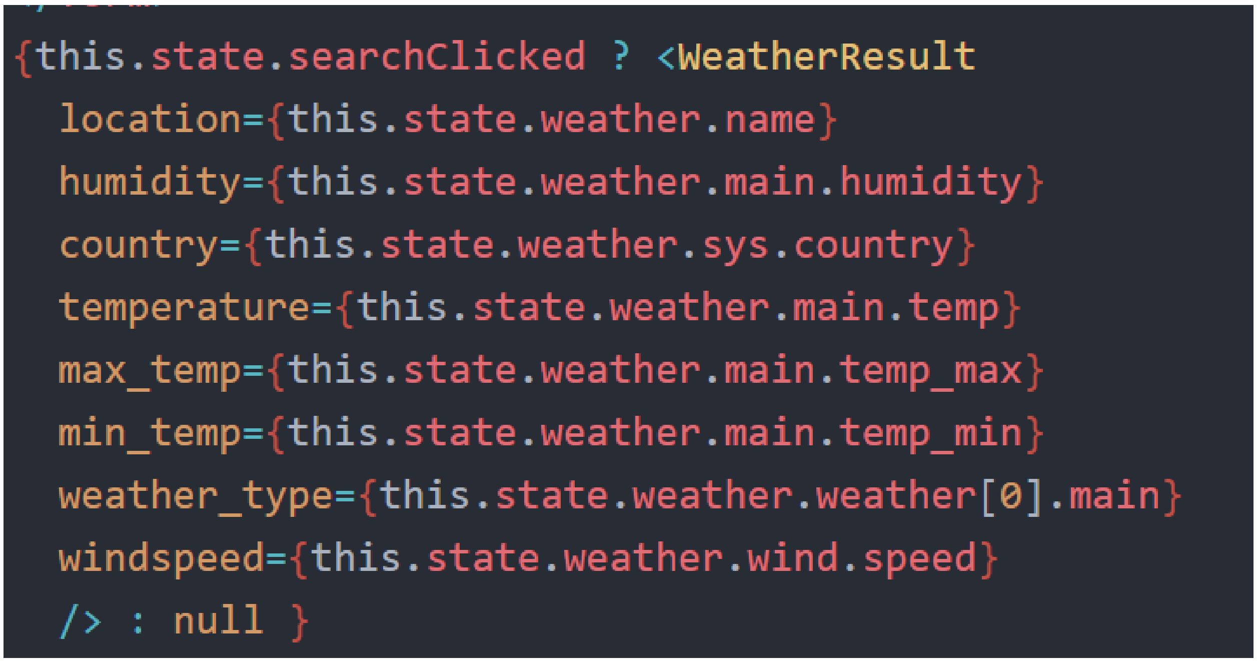
This component shows the user the current weather status from a desired place. At the beginning the <WeatherResult /> is hidden as it contains no properties. Once a city is written in the form field, a check is run to verify if the city is in the database of the API, this is done using a filter function applied to a json filed already existing on the server machine.



```
let elementPos = cityList.map(function(x) {return x.name.toLowerCase(); }).indexOf(event.target.value.toLowerCase());
let objectFound = cityList[elementPos];
if(objectFound){
  this.setState({showErrorLabel:true, inputMessage:"This city is in our database", labelClass:"text-success"});
}else{
  this.setState({showErrorLabel:true, inputMessage:"This city is not in our database",  labelClass:"text-danger"});
}
if(event.target.value === ""){
    this.setState({showErrorLabel:false});
}
```

Figure 1 Filtering the user input

If the city is valid a request is sent the database using the “axios” library. This library allows the use of an API without using any server-side code. Once the response is received the data is sent onto the <WeatherResult /> as properties, see *Figure 2 Calling Component using properties.*

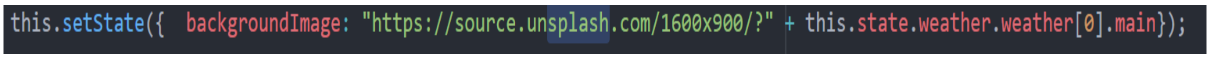


```
{this.state.searchClicked ? <WeatherResult
  location={this.state.weather.name}
  humidity={this.state.weather.main.humidity}
  country={this.state.weather.sys.country}
  temperature={this.state.weather.main.temp}
  max_temp={this.state.weather.main.temp_max}
  min_temp={this.state.weather.main.temp_min}
  weather_type={this.state.weather.weather[0].main}
  windspeed={this.state.weather.wind.speed}
  /> : null }
```

Figure 2 Calling Component using properties

The <WeatherResult /> is a functional component and doesn't have a state, as it doesn't require one. Its purpose is to only display the properties it has every time it gets new ones.

The picture on the left side is also changing depending on the data received. The weather condition is used as a filter option for a request to the Unsplash API. This API is a giant library that offer free to use images.



```
this.setState({  backgroundImage: "https://source.unsplash.com/1600x900/?" + this.state.weather.weather[0].main});
```

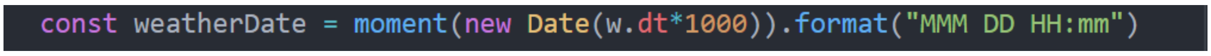
Figure 3 Using Unsplash API

FutureWeather.js and Graph.js

These two components handle the second functionality of the application, which is allowing the user to view the weather in the future days. Open Weather offer a different API call showing the weather for the next 5 days, with a result for every 3 hours.

The same filtering option is used for the form field in order to let the user input only cities that are in the database. Once the request is made, an array of 40 entries is received. The list is then sent as a property to the <Graph /> component. The graph displayed is created using the Chart.js library, which seem to be the best solution after experimenting with other react and non-react libraries, like "d3" and "react-charts".

It is worth mentioning that the bottom axis of the chart was parsed using the "moment.js" library. This library is capable of parsing full date formats into whatever the user needs them, in this case, 3 letters of the month, the day and the hours and minutes.



```
const weatherDate = moment(new Date(w.dt*1000)).format("MMM DD HH:mm")
```

Figure 4 Parsing date with moment.js

Once the <Graph /> is rendered two more functionalities are added to the screen. One allows the user to change the style of the graph (from line chart to bar chart) and one filters the data used, by shortening the dataset used to build the chart.

Styling

The focus of the project was to make all the functionality work, so the style wasn't given much attention. This project uses a similar style found on a video tutorial. The application is adapted and might look similar, but the core is different.

In short terms, there is a box in the middle of the screen that holds all the contents of the application. The navigation bar is on top of the box. When a link is clicked the class of the link changes giving the border a different colour. A similar approach is used when showing the label for the city validation. Even if the contents of the box are aligned using the bootstrap framework, the form and the text is styled differently so it matches with the rest of the colours. There are three google fonts used on this project: Merriweather, Roboto and Open Sans.

Type of users

The purpose of this application is experimenting with React and libraries and might not be the most wanted application and most of the people would probably not find a reason in using it as there are better and faster ways to check for weather results.

Even if the app might not be in demand these are some examples of users that might show some interest in this:

1. As a farmer in Ireland I must always check the weather conditions to prepare for any weather conditions and this app offers an easy to understand graph that shows temperatures every 3 hours.
2. As a person that works night shifts in a place that doesn't offer any view to the outside, I want to know what kind of weather is waiting for me when I finish my shift in the morning.
3. As a data analyser I always want to know how the weather temperature differences from a place to another and this app offers a complete graph of the following 5 days with data entries every 3 hours.
4. I work in a big hall and sometimes I hear noises on the roof. I immediately check the weather conditions to see if it is raining or there is a problem with the roof.

Reflections on the project

By working on this project, I learned most of the basics of react, like: different ways to create a component, how does it a component work, the reason to use a state or stateless component, how to change states, how to invoke different components using a value (conditional rendering), routing and linking. I also got to experiment with other libraries (moment.js, chart.js, axios).

The biggest struggle was to figure out how to work around async functions as most of my components wouldn't re-render or would crash just because the state would take longer to set.

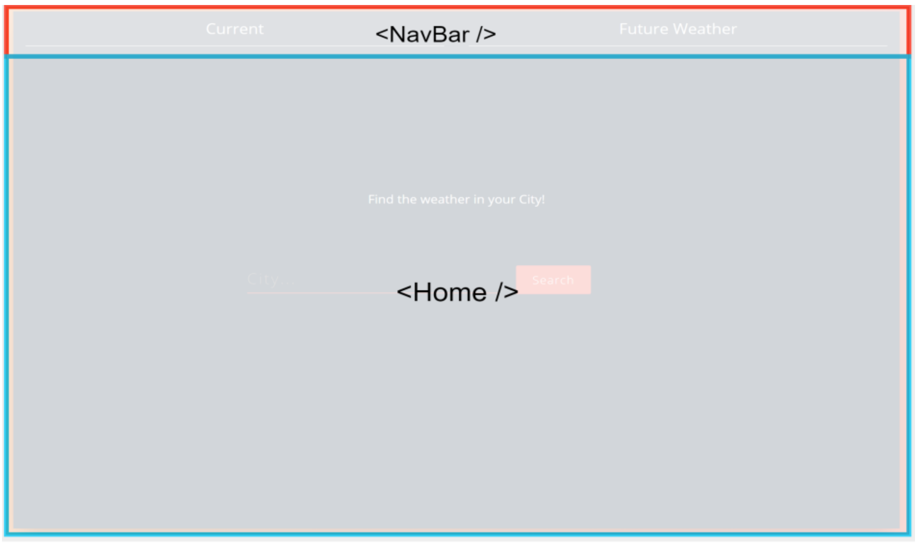
If I had dedicated more time to this project I would have fixed all issues and would make the application more responsible using React Native for the phone users and use Context to store the variables I had to redeclare every time I needed them (API_KEY, KELVIN, cityList).

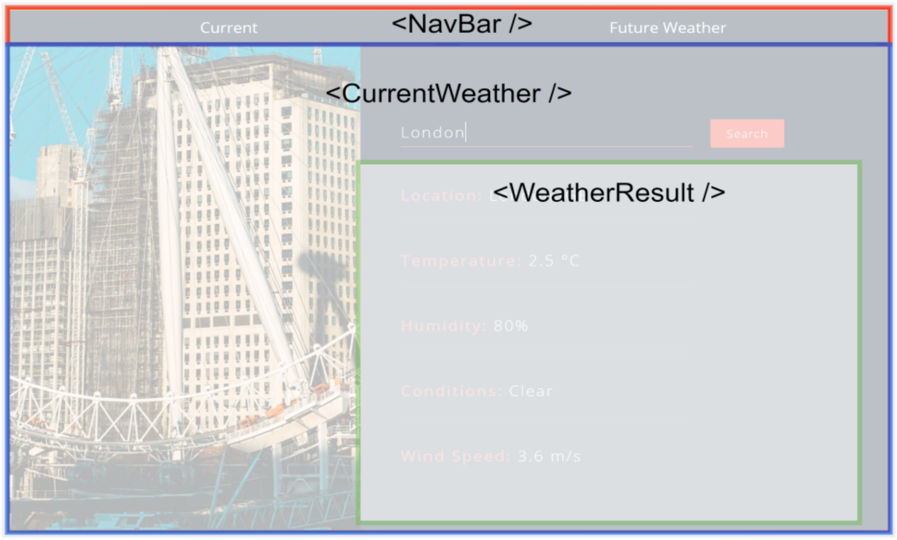
In conclusion, this was a fun project to work on and I have learned a lot from it and now that I am more experienced I can't wait to design a new one.

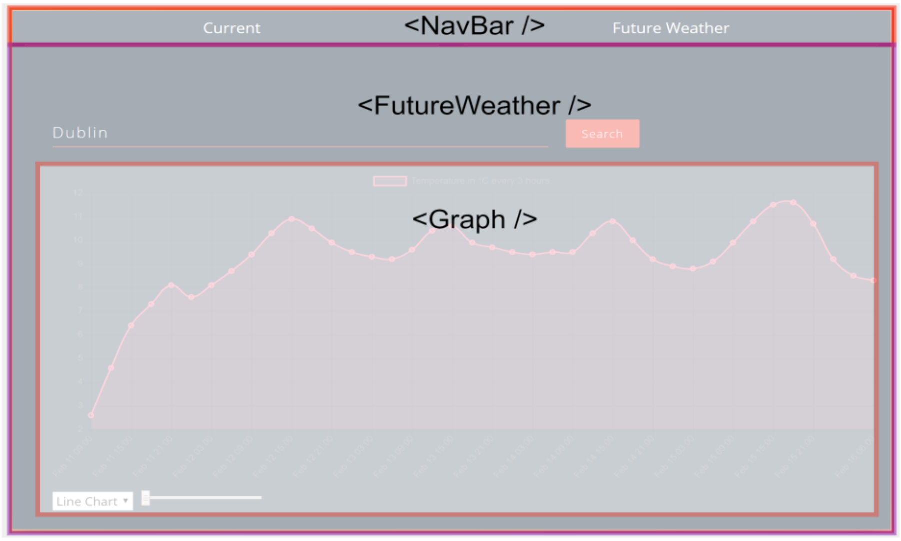
Github link

https://github.com/georgeBl/Advanced_JS_CA1

Appendix A Wireframes


Current
<NavBar />
Future Weather
Find the weather in your City!
City
<Home />
Search


Current
<NavBar />
Future Weather
<CurrentWeather />
London
Search
<WeatherResult />
Location:
Temperature: 2.5 °C
Humidity: 80%
Conditions: Clear
Wind Speed: 3.6 m/s


Current
<NavBar />
Future Weather
<FutureWeather />
Dublin
Search
<Graph />
Line Chart