Oregon State University

CS 461

Fall 2017

Tech Review: Linking Seasonal Weather Data to AgBizClimate

Author:
Shane Barrantes, 29

Instructor:
D. Kevin McGrath
Kirsten Winters

Abstract

This document will provide an overarching analysis on front-end frameworks, graphing frameworks, and back-end design that we considered and selected for the AgBizClimate project. The topics included in this document will mirror my primary responsibilities for this project.

1 Front-End Frameworks

1.1 Overview

Front-end technologies are extremely important to web applications because they supply the foundation and interface for user interaction. This interaction is the first glimpse that users get into the application and will use it to judge the appearance, feel, and usability of the product. We want to use a front-end framework to give structure to our front-end interface, creating a groundwork for our application and making sure we can add any additional features with minimal time expenditure. We will be using the framework to build a user interface that is easily navigable and will provide dynamic graphing output based on user input.

1.2 Criteria

The front-end framework we pick needs to accomplish several things. First, it must enable our project to be highly customizable. Secondly, it needs to have a logical structure that allows us to create new features easily and produce a functional interface. Lastly, it needs to play nicely with graphing libraries so that we can cleanly display the graphing output we generate from the seasonal weather data to the user.

1.3 Potential Choices

1.3.1 Angular JavaScript

Angular JavaScript is a relatively new web framework built around HTML5, CSS3, and JavaScript that is developed and maintained by Google[1]. Angular JavaScript is not Googles first framework, but it is the most modern so it is highly dependable. Since the debut of this framework it has standardized the way web applications are structured and is used as a model for front-end design[1]. Angular supports easy REST actions, Model View Controller, and Model View View-Model.

1.3.2 React JavaScript

React is the latest and greatest front-end JavaScript library. It is only four years old and quickly becoming the most used front-end technology due to its simplicity and strength in building user interfaces[2]. This strength comes from the fact that you dont have to write separate HTML with react, but instead you describe what you want and React builds the HTML for the designer. Perhaps Reacts biggest strength is its reactive updating. When an input is changed React instantly updates the component without refreshing the page.

1.3.3 Raw JavaScript

JavaScript without additional frameworks is still fully functional for designing a front-end interface and structure and brings several strengths. The first strength is that raw JavaScript designers dont have to spend additional time learning and choosing which framework they want to use. Secondly, getting off the ground and creating the application can be easier due to not having to spend time setting up the application framework which can sometimes be overkill in comparison with the project goals. Lastly expanding the code base once its created will be easier since you wont have to pull in and learn additional frameworks, but this strength only exists if creator is also the maintainer of the wrote the code.

1.4 Discussion

Raw JavaScript's benefit is that we wont have to learn additional frameworks and can immediately write our own code base. However, for this project I believe it is a weakness since the rest of the AgBizTeam will have to maintain our code after we complete the module; so existing within a standardized framework is a good idea. The existing model for AgBizClimate is model view controller which Angular JavaScript was built to support therefore making the product is consistent and reliable. React is a phenomenal front-end technology with no real weaknesses except for not being a self contained framework.

1.5 Conclusion

For the purposes of this project raw JavaScript is not a viable solution for the previously mentioned reasons. React JavaScript and Angular JavaScript dont play well together so its important that we only use one of them for the front-end technology. Since our team is producing a submodule of an existing product we are required to use the technologies the development chose at the start of development; so we will be using Angular JavaScript for our front-end framework. As a side note we will also be using the front-end HTML framework Bootstrap with React.

2 Graphing Frameworks

2.1 Overview

One of the essential and most valuable components of the AgBizClimate submodule is its ability to generate useful and explicit graphs based on user input. These graphs need to be visually pleasing and easy to interpret so that users can quickly ascertain the relevant information they can use to assist their normal decision making process.

2.2 Criteria

The graphing framework that we choose needs to work closely with JavaScript and HTML5 since that is the primary base for our dynamic web application. It also needs to work quickly since we are trying to provide user with rapid feedback based on input and have high responsiveness and feedback to increase overall user satisfaction. Ideally the graph selection we chose will have minimal overhead so integration is quick and easy.

2.3 Potential Choices

2.3.1 Angular Chart.js

Angular chart.js is an open source graphing library for JavaScript. It is responsive and works well with JavaScript and HTML5. It can provide eight different types of charts and can interleave different chart types together [4]. This interleaving process creates singular graphs that can illustrate data differences more clearly. Chart.js is script-able and also supports animations which could assist users in understanding the data[4].

2.3.2 Plotly.js

Plotly.js is a high level JavaScript graphing library that is built on top of d3.js and stack.gl[5]. It provides the ability to chart data with 20 different chart types including 3D graphing, animations, sub-plotting, and mixed plotting[5]. One of the stand out aspects of Plotly.js is its ability to stream data in and dynamically produce graphs.

2.3.3 D3.js

D3.js for data driven documents is a JavaScript library that was created solely to manipulate documents based on data. It is one of the most widely used and popular JavaScript graphing libraries in existence. It works closely with front-end frameworks to produce high quality HTML5 tables and visualizations. D3 is sleek, fast, and effective; allowing high quality graphs to be generated with almost no overhead and assisting in high responsiveness and usability[6].

2.4 Discussion

Chart.JS is an effective open source JavaScript library that would work well for our project. Its primary drawback is the limited number of chart types. With the current AgBizClimate setup this is not an issue, but when the submodule expands it could create problems down the road due to limitations on visualizations. Plotly.js is an extremely powerful JavaScript tool built on top of the other option, D3.js and provides a wide variety of graphing options. However, due to the size and high level of abstraction Plotly.js is slower than the other two graphing libraries with more required overhead. D3.js is the nice middle ground between these three libraries. It's faster with more responsiveness than plotly.js and it provides a wider range of chart types than Charts.js.

2.5 Conclusion

All of the graphing frameworks this document has discussed have their varying strengths and weaknesses, however I believe D3.js is best suited for the job. Since our team is producing a submodule of an existing product we are required to use the technologies the development chose at the start of development; so we will be using Angular Chart.js for our graphing framework.

3 Back-end Designs

3.1 Overview

The Back-end design or software architectural pattern is an essential part of building a functioning web application. Deciding on which model to use will influence design decisions for all parts of the web application including the front-end, and the back-end The following choices for back-end design are the most popular and modern options that are being used today.

3.2 Criteria

The back-end technology has two key requirements in order for it to be successful with the AgBizClimate project. The first requirement is that there needs to be a distinct front-end and back-end. Secondly, it is essential that the front-end is stated and sessioned, so that the application remembers the user inputted steps to reach the decision assistance stage.

3.3 Potential Choices

3.3.1 REST API

REST APIs also known as a representational state transfer application passing interfaces have a clear separation between the client and server which is highly desirable for creating seamless user interfaces[7]. This separation makes products using REST extremely scalable with very little effort. Rest APIs are also useful due to the API itself being separate from the code-base. This means that you can have servers running different languages, but as long as the API is the same then the application will still function.

3.3.2 Model View Controller

The Model View Controller is one of the most basic and widely used back-end architectural patterns. The Model consists of the logic and collection of classes necessary for the web application. The View is what the user sees and where the user interface resides and the controller is what handles requests and passes information between the model and the view[9]. The Model View Controller Paradigms main strength is the separation between the visual components of a web application and the functional back-end. This allows the front-end interface to be altered with no real effect on back-end functionality.

3.3.3 Model view ViewModel

The Model View View-model design pattern is utilized to separate the front-end from the back-end with an integrated ViewModel component. The Model consists of the logic and collection of classes necessary for the web application. The View is what the user sees and where the user interface resides. The ViewModel is responsible for altering the state of the view and manipulating the model with the information that was gained from the altered view[8]. This paradigm allows events to trigger in the view itself.

3.4 Discussion

Each of the previous paradigms have their strengths and weaknesses. The REST API is fantastic for simple queries to the back-end, but it has a harder time tracking progression through multiple steps which is what we need for AgBizClimate. The Model View ViewModel method is great for projects that have long forums and require more dynamic views, but its a bit overboard for our needs on this project. The Model View Controller method allows the designer to have a distinct front-end and back-end while processing data between the components. Maintaining this structures allows the development team to spend very little time working on the front end after it's initial design and completion.

3.5 Conclusion

Since our team is producing a submodule of an existing product we are required to use the technologies the development chose at the start of development; so we will be using the Model View Controller paradigm for our back-end design.

References

[1] Angular JavaScript
https://www.linkedin.com/pulse/20140613173601-45832080-why-to-choose-angularjs-javascript-framewo
Pankaj Kumar Jha, June 13th, 2014.

[2] React JavaScript,
https://medium.freecodecamp.org/yes-react-is-taking-over-front-end-development-the-question-is-wh
Samer Buna March 30th, 2017.

[3] Raw JavaScript,
https://www.sitepoint.com/frameworkless-javascript/. Pawel Zagrobelny.

[4] Chart.js
http://www.chartjs.org/.

[5] Plotly.js,
https://plot.ly/javascript/.

[6] d3js,
https://d3js.org/.

[7] The Representational State Transfer,
https://www.service-architecture.com/articles Douglas K Barry.

[8] The MVVM Pattern,
https://msdn.microsoft.com/en-us/library/hh848246.aspx

[9] The MVC Pattern,
https://msdn.microsoft.com/en-us/library/windows/hardware/ff550694(v=vs.85).aspx