Oregon State University

CS 461

Fall 2017

Tech Review AgBizClimate©

Author:
Thomas Noelcke

Instructor:
D. Kevin McGrath
Kirsten Winters

Abstract

The purpose of this document is to research and consider different technical options for our application. In this document we research different options for data storage, HTTP request frame works, and testing frameworks. We will consider three possible choices for each section of the application. For each of these options we will weight the pros and cons of each. After comparing the different options we will select the option we would like to use for the *AgBizClimate* application. We have divided our application into 9 different section and each of us has preformed this analysis for each of the nine sections.

Contents

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

Oregon State University

CS 461

Fall 2017

Tech Review AgBizClimate©

Author:
Thomas Noelcke

Instructor:
D. Kevin McGrath
Kirsten Winters

Abstract

The purpose of this document is to research and consider different technical options for our application. In this document we research different options for data storage, HTTP request frame works, and testing frameworks. I will consider three possible choices for each section of the application. For each of these options I will weight the pros and cons of each. After comparing the different options I will select the option I would like to use for the *AgBizClimate* application.

Contents

1 Data Storage

1.1 Overview

For the *AgBizClimate* application we will need a way to store data so it can be easily retrieved later. For this application we will store a variety of data including budget data, weather data, and user information. This information will need to be quickly recalled so it can be used in our application. Generally, we will want to select a data storage option that will be easy to set up and allow us a lot of flexibility with what kind of data that can be stored. We will also want a data storage option that will quickly recall stored data so it can be used by the application.

1.2 Criteria

To determine the best choice for our application i will analyze the performance of the data storage based on, Ease of Development, and Ease of Set Up and Flexibility. I will analyze run time speed however, this will not be one of the criteria as it is not critical that our data is retrieved as quickly as possible. Generally, its much more important to consider Ease of development, Set up and flexibility because the primary concern is being able to get the application up and running quickly. Ease of development and ease of set up will be a subjective measures for each option. In those sections I will use the opinion of other software developers along with my own experience to compare each option.For these criteria I will rate each option on a scale from very easy to very hard. Flexibility will be a measure of how easy it is to store different kinds of data using each option. For Flexibility I will measure each option from flexible to rigid.

1.3 Potential Choices

1.3.1 PostGreSQL

PostGreSQL is an open-source relational database management system. PostGreSQL uses the Sever Querying Language (SQL). PostGreSQL uses the relational database model. This model sets up tables that represent a certain type of data. We can then run SQL quires on this data base to get the data we need for our application [1].
Though SQL is quick, In my experience it is not as developer friendly as the other options in this analysis. Generally, writing raw SQL queries is difficult and time consuming. This is especially true when your data models are very complicated. Using Raw SQL also requires the developer to manually figure out how to make the mapping between the database and the models used at the application level. With raw SQL it is also more difficult to change the structure of the database once you have created the database. These changes will require potentially complicated scripts along with scripts that convert the data to fit into the new structure [2].
Another problem with SQL in my experience is that it is more difficult to set up. The configuration process can be complicated. Additionally, if you choose to use raw SQL you must also set up your data base as third party application separately from your actual application.
Another problem with SQL is that it is rather rigid in the way that you must store data. For example if you want to store a list object in an SQL data base you must create a new table where one row represents one item in the list. This item must have its own unique id. Additionally, if you want to have a list of lists it gets even more complicated. Now you need to create another table to represent a name and ID for each list you want to store and you must relate every item you want to put in that list back to the parent item in another table. This can get very complicated in a hurry. Another problem is if you don't know what the structure of the data is going to look like before run time it is impossible to store this data in an SQL database. This makes PostGreSQL rather rigid in

terms of flexibility.

1.3.2 Python ORM

Python ORM or Object Relational Mapper does not substitute for an SQL database. There will still need to be an SQL database running on the back end. However, the ORM framework allows developers to create objects in python that then map to the data base. Often times this type of frame work allows the developer to create the objects first and let the framework deal with creating the SQL database on the backend. Given that this is not a replacement for an SQL it does make working with an SQL database much easier.
This type of frame work has several advantages, it allows for easy development and set up. The ORM frame work allows the developer to be completely insulated from the SQL data base on the back end. This means that instead of writing SQL queries to get data from the database the developer is able to use objects in python to access data that is stored in a database. This makes development and set up Easier on the developer. This is because the developer doesn't have to worry about writing complicated SQL statements or setting up complicated relationships between tables. This frame work also allows the developer to develop the code first and let the framework worry about creating the database the data will ultimately be stored in [3]
The ORM framework also allows for great flexibility in what you can store in a database. This type of frame work allows for mappings between python objects and the data base. So nearly anything you can store in an object in python, you can also store in a database. However, it should be noted that you must know the structure the object you are trying to store before run time.
Though this approach is very easy for the developer it isn't with out cost. The ORM approach does take a hit in terms of run time performance. The ORM framework will be slower than raw SQL. Another problem with this type of framework is that it doesnt leave the developer very much control over the database. This means that you are stuck with what you get. If the frame work structures something in the database in a way you don't like you don't have a ton of choice about that. Additionally, making database chances can cause you to loose data in the database if you are not careful about how you handle the migrations [3].

1.3.3 MongoDB

MongoDB is an open source database NoSQL database. This means that instead of using tables and rows like an SQL relational database, MongoDB uses collections and documents. Documents are individual data members where there is a key value associated with each document. Collections contain multiple documents. Collections allow the developer to store many items in a database. This structure allows for the structure of the data being stored to be determined after run time. This allows for greater flexibility because you can store dynamically generated objects[4].
MongoDB is also fairly simple to set up. This is because of the way that objects are stored we don't need to set up and complicated tables. We can simply set up our data base and insert the data. This also means that if we want to change the structure of the data down the road we don't need to make massive changes to the data base. This makes MongoDB much more developer friendly and also very flexible[2].
However, It should be noted that the usability and flexibility of MongoDB is not with out cost. The biggest cost of the flexibility of MongoDB is that it is harder to relate two items in a database. For instance if want to have one entity that represents a user and another that represents a user role, this becomes a challenge in MongoDB. There are work around and ways to solve this problem however it should be mentioned that this is a problem[5]. Another trade off in using MongoDB read and write operations are generally asynchronous. This means that you can tell the database to do something and then move on other tasks and the data base will return the result of your query later.

On the surface this sounds nice but if you are doing a lot of reading and writing operations in the same block of code this can cause problems. For instance if you write several items to the data base and then need to read items out that depend on those items you just added you may get errors because the first write hasn't finished yet[6]. Finally, MongoDB may also be much slower for some operations than traditional SQL. One example where this becomes apparent are aggregate functions where you want to preform some sort of manipulation of the order of the data[7].

1.4 Discussion

As mentioned earlier we can divide the types of data we need to store into two distinct groups, weather data and user data. The user data will generally have relationships between different parts of the data. For instance a user will have an address and a phone number. The user will also have various different budgets. These relationships make the data easier to store in an SQL database such as PostGreSQL. To farther simplify storing the data we could also use Python ORM. This would allow us to create relationships in the data, easily develop the database and store the data. The weather data on the other hand is much more like a large list. This sort of data can be sotred in an SQL data base such as PostGreSQL however, this would make the development of the project more difficult. A more suitable choice for storing the weather data would be MongoDB. This is because MongoDB allows for easy storage of lists with little complexity than PostGreSQL. MongoDB provides the easiest and most flexible way to store the weather data.

1.5 Conclusion

For this project we will use PostGreSQL and MongoDB. We decided to use these frame works because our client requested it. This is because we are adding to our clients system that has already been created. However, the client did put in some good though into these choices. They chose PostGreSQL because its a powerful relation database management system that will work well for the user data. They also chose MongoDB for the weather data because it provides a simple to set up and easy to use storage tool for storing large lists of data.

2 HTTP Request FrameWork

2.1 Overview

The *AgBizClimate* project will need a way for the client on the front end of the application to communicate with the API at the back end of the application. We have already determined that we will use HTTP request to facilitate this communication. However, we have not decided which framework we should use to make the HTTP requests with. Generally, the frame work we use to make HTTP requests should be easy to use, easy to read and allow us to quickly right requests.

2.2 Criteria

For the purposes of this analysis we will consider ease of development, readability and compatibility with AngularJS. We are considering compatibility with AngularJS because our client has required us to use angular. This will be measured on a scale from compatible to uncompilable. Ease of development will be a subjective measure of how hard it will be to develop software with a framework. This will be measured on a scale from easy to very hard. Readability will be a subjective measure

of how easy it is to decipher the meaning of the request from the code. This shall be measured on a scale from readable to incomprehensible.

2.3 Potential Choices

2.3.1 jQuery Ajax

When talking about HTTP requests in java script Ajax is probably what immediately comes to most software developers minds. Ajax is one of the original frameworks for making HTTP requests with out updating the page. Ajax allows for asynchronous requests to the backend. Generally, Ajax is not very human readable. Ajax can be read by humans but those humans are going to need to have had experience with Ajax before. It is important to note that Ajax requests will require more work that most of the other HTTP request frameworks we are discussing. Additionally, Ajax can be use with Angular however we will be required to take extra steps to ensure that the requests are handled correctly. If you choose to use this framework with angular you will also need to do more work to be able to test your code.

2.3.2 axios

Axios is a promise based HTTP client for web browsers. Axios will also allow us to make asynchronous calls to the backend of our application. It is also important to note that axios HTTP requests are much more concise making them human readable[8]. Axios can also be integrated into Angular however, this will require some set up. If you choose to use axios with angular you will need to set up your own custom tests because this frame work will not integrate into angular testing framework. Additionally, you may run into headaches down the road when angular updates are made because axios is not maintained by angular.

2.3.3 AngularJS $http

$http is a module in AngularJS that enables communication with remote HTTP servers by using the browsers built in tools. This frame work provides a layer over top of an ajax request that makes the request easier to read and write. These requests are much easier to read and writing the request is much easier that raw Ajax. This HTTP frame work also integrates seamlessly with AngularJS because its a core service. This means that testing these requests with the Angular testing frame work is fairly easy and updates to Angular wont break your HTTP requests [8][9].

2.4 Discussion

In comparing these three frame works I noticed that $http really stood out from the other two frame works. This is because $http is a core service in the AngularJS framework. This makes it much more compatible with Angular than Ajax or axios. Comparing $http and axios out side of the context of Angular they are pretty similar. Both produce much more readable http requests than Ajax and both are are fairly easy to use.

2.5 Conclusion

For this project we will use $http to make http requests between the front end of our application and the back end of our application. We chose to use this framework because it integrates seamlessly with Angular while also making HTTP request readable and easy to develop.

3 Testing Frame Work

3.1 Overview

For the AgBizCliate system we will be required to write unit tests for our code. Unit tests will help us ensure that our application meats the functional requirements. Unit tests will also be helpful for feature projects to ensure that parts of the application have not been broken by a change. In general good unit tests will improve the reliability and extend the life of a project. For this project we want a testing frame work that will allow us to quickly create unit tests in Python for Django Applications. We want to be able to do this because our client has already specified that we will be using Python and Django.

3.2 Criteria

For the purposes of this analysis we will consider ease of development, compatibility with Django, and simplicity of setup. Ease of development is a subjective measure of how easily tests can be developed with each framework. Ease of development will be rated on a scale of easy to hard. This frame work will also need to be compatible with Django as this is the frame work our application will be written in. Finally, we will also consider ease of set up. This is a measure of how easy a framework is to set up so we can begin writing unit tests.

3.3 Potential Choices

3.3.1 Python unittest

Python unittest is a testing frame work built into Pythons standard library. The unittest framework provides the necessary scaffolding to setup, shutdown and run unit tests. This testing framework will feel familiar is it is part of the greater Junit project. This makes writing unit tests feel familiar and fairly easy. However, for this application using this testing frame work would require a lot of set up. This is because this is a web application and unittest does not include tools that make it easier to set up and run tests for web applications. More generally, unittest wills take longer to develop because it provides less scaffolding[10]. This makes unittest less developer friendly. Unit test can easily be used with Django as it is part of the python library. For this reason unittest is also very simple to set up.

3.3.2 py.test

py.test is a popular python testing framework that helps reduce the repletion required in unittest. py.test also contains powerful utilities that make testing many kinds of applications easier. For our project py.test would make testing our web app much simpler than unittest. This would make test development quick and easy. Though this framework makes tests easier to develop that does come at a cost. This framework requires extra setup to integrate with Django. This will make setting up this framework more difficult that either Django.test or unittest. Though the set up isn't complicated it would make testing our application more difficult and more complicated [11].

3.3.3 Django.test

Django.test is a testing frame work that is baked into Django. This frame work provides powerful tools for testing Django applications. This frame work is part of the greater Django framework. Because Django.test is built into the Django framework, there is very little set up. This also means that there will be no compatibility issues with Django either now or in the future. This testing suite

also provides powerful utilities that make it easier to develop tests. This frame work also gives us specific tools designed to test Django applications[13].

3.4 Discussion

While considering the options there is one option that fits our application the best which is Django.test. Django.test will allow us to test our Django application with minimal effort as it requires no special set up. Additionally, Django.test also contains tools like py.test that will enable quick and easy development of tests. Though py.test may also be a good option, it doesn't integrate into Django as well as Django.test. Unittest is also another good testing option as it is apart of the python language. However for the purposes of this project unittest would make the development of tests more difficult.

3.5 Conclusion

For this project we will be using Django.test to write unit tests for our application. We chose to do this in part because our client asked us to use this frame work. However, our client asked us to use this frame work with the goals for our application in mind. For this application we want to be able to quickly and easily generate unit tests and Django.test is one of the best tools available for testing Django applications.

4 References

[1] Home, PostgreSQL Tutorial. [Online]. Available: http://www.postgresqltutorial.com/what-is-postgresql/. [Accessed: 22-Nov-2017].
[2] MongoDB vs SQL: Day 1-2, MongoDB. [Online]. Available: https://www.mongodb.com/blog/post/mongodb-vs-sql-day-1-2. [Accessed: 22-Nov-2017].
[3] M. Makai, Object-relational mappers (ORMs), Object-relational Mappers (ORMs) - Full Stack Python. [Online]. Available: https://www.fullstackpython.com/object-relational-mappers-orms.html. [Accessed: 22-Nov-2017].
[4] What is MongoDB? - Definition from WhatIs.com, SearchDataManagement. [Online]. Available: http://searchdatamanagement.techtarget.com/definition/MongoDB. [Accessed: 22-Nov-2017].
[5] J. Headley, The Problem with MongoDB Hacker Noon, Hacker Noon, 12-Feb-2017. [Online]. Available: https://hackernoon.com/the-problem-with-mongodb-d255e897b4b. [Accessed: 22-Nov-2017].
[6] T. S. Chief, Potential problems and issues with using MongoDB, Stackchief. [Online]. Available: https://www.stackchief.com/blog/Problems%20with%20MongoDB. [Accessed: 22-Nov-2017].
[7] A. C. Weinberger, Benchmark: PostgreSQL, MongoDB, Neo4j, OrientDB and ArangoDB, ArangoDB, 13-Oct-2017. [Online]. Available: https://www.arangodb.com/2015/10/benchmark-postgresql-mongodb-arangodb/. [Accessed: 22-Nov-2017].
[8] axios, npm. [Online]. Available: https://www.npmjs.com/package/axios. [Accessed: 22-Nov-2017].
[9] $http, AngularJS. [Online]. Available: https://docs.angularjs.org/api/ng/service/$http. [Accessed: 22-Nov-2017].
[10] unittest vs py.test, Bytes IT Community. [Online]. Available: https://bytes.com/topic/python/answers/43330-unittest-vs-py-test. [Accessed: 22-Nov-2017].
[11] 26.4. unittest - Unit testing framework 26.4. unittest - Unit testing framework Python 3.6.3 documentation. [Online]. Available: https://docs.python.org/3/library/unittest.html. [Accessed: 22-Nov-2017].
[12] C. Maske, The Engine Room, Using pytest with Django - The Engine Room - TrackMaven. [Online]. Available: http://engineroom.trackmaven.com/blog/using-pytest-with-django/. [Accessed:

22-Nov-2017].
[13] Django Tutorial Part 10: Testing a Django web application, Mozilla Developer Network. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Testing. [Accessed: 22-Nov-2017].

Oregon State University

CS 461

Fall 2017

Tech Review

Author:
Shengpei Yuan

Instructor:
D. Kevin McGrath
Kirsten Winters

Abstract

The purpose of this document is to research different technical options and consider possible choices for our application. In this document, I have researched different options for containers, back end design, and styling framework, and I will analyze three choices for each section of the application. For each option, I will introduce the principle, advantages and disadvantages as it relates to this project. After comparing all three choices, I will choose the appropriate option for the AgBizClimate application.

1 Container

1.1 Overview

Containers are a powerful virtualization technology that is helpful for software development with the cloud environment. Generally speaking, they greatly help environment construction and deployment work for developers and maintainers. Containers also improve efficiency and reduce cost of developing complicated software systems.

1.2 Criteria

Containers are one important technology in cloud computing. It provides independent running environment for various software applications. It is believed that containers may replace virtual machines in the near future as the dominant technology for constructing cloud-computing environments. Basically, containers work on the OS layer and provide runtime environments for programs. The OS distributes computing, memory and peripheral resources for containers, and it owns isolated namespaces for programs.
The performance of containers should be considered carefully as there are substantial differences between computing architectures and their native hosts. Ideally, we would expect little or no reduction on computing or memory access efficiency when using containers relative to the native host. To test whether containers meets this requirement, one feasible method is to examine and compare the specifics of the same application deploying both on containers and native host.

1.3 Potential Choices

1.3.1 LXC

LXC is one of the earliest containers to provide virtual Linux runtime environment for software applications. The core characteristics of LXC are built on the basis of resource management and isolations techniques of the Linux kernel. For instance, the cgroups feature of Linux kernel that divides processes into groups for management is the essential framework for managing resources for processes in LXC.

1.3.2 Docker

Docker is a mature container tool has almost replaced the conventional Linux Containers (LXC) and helped solve the availability problems of many software systems. It is claimed that Docker is the prevailing virtualization technology in todays software building environment. However, with the rapid development of the Docker ecosystem as a tool for cloud-computing it has made constructing other businesses based on it harder. Docker is one of the best components for constructing a software system in virtual computing environments. One perspective is that Docker is or will become an ecosystem for software construction. This will make it vulnerable in many ways, whereas Rocket makes a fresh start and only focus on working as a component to help construct complex software systems. Another perspective is that the core value of Rocket is exactly the start point of Docker, integrating complicated software systems into one independent platform. The founder of Docker, Hykes agrees that Rocket could be one important tool for customizing configurations of containers, and will define the future of containers technology.

1.3.3 Rocket

Rocket is a quite new open-source container technology for software developing created by CoreOS in 2014. It is implemented in Go. As a command-line tool, Rocket works quite like Docker to package software applications and their dependency libraries or systems to a plantable container.

It is known that Rocket would have no compatibility issues with Docker. Also, rocket will bring new ideas for software container technology and relevant markets. It never tries to provide broad friendly functionalities like cloud acceleration tools, and integrated systems. Rocket will become more purified as a standard container tool for the software industry. It has a start point that multiple heterogeneous container platforms could be integrated into. This will help solve the availability problem. Moreover, it would try to alleviate the security issues of Docker. It should be noted that Docker and Rocket both have their own strength and weakness. Generally speaking, Docker would be more competent at complicated and huge systems, while Rocket is a better fit for lightweight and small software applications.

1.4 Conclusion

The vast adoption of Rocket among great software systems proves that it satisfies the requirements of the software world. Rocket is designed to boost the success of software systems, including specific measures like guaranteed security, flexible components and open standards. Different companies may choose different container tools between Docker and Rocket according to the specific contexts of their systems. As a result, for this project, we will adopt Rocket as an independent container library, so that it could be easily integrated into the existing project AgBizClimate and the corresponding environment. In other words, we need a container tool to construct components for our system rather than to create a new system from scratch.

2 Back end Design

2.1 Overview

There are several powerful and popular models for building the back end of a web applications. RESTAPI and MVC are two important technologies among them. Besides, we would also describe a little about MMVC, an extended version of MVC. We will introduce them in the order of when they were proposed in order to help readers to understand them more easily.

2.2 Criteria

The backend architecture technologies mainly solve the problem of decomposing the functionality of the application reasonably and efficiently for software developers. Ideally, all modules have clear bounds between each other and tight connections within themselves. Besides this, it is also important to present the structure and functionality of the application clearly and easily to both developers and end-users.
The performance of the back end architecture is important as once determined they dominate the entire development process of the application. The back end architecture has great impact on the extendability and robustness of the program. It can be hard to test whether a back-end architecture can fully meet our design requirements as it is very abstract high layer design. However, it is commonly believed that a good back-end architecture must make improvements on the efficiency of entire process of system development, function extensibility, code readability, and maintainability.

2.3 Potential Choices

2.3.1 MVC

The MVC (Model View Controller) model was proposed as early as 1970s and used by Smalltalk, and it is still popular and used to develop large software applications in various fields today. Many software languages and frameworks like the prevailing Java Struts and Spring MVC use the MVC architecture. The core concepts behind the MVC model is that no matter how simple or complex a

software system is, it could always be stratified into three layers from the perspective of structure. First, the View layer, which lies on the top, is the one directly seen by the end user. It usually works as an interface between users and the program, and is also called the shell of the program. Secondly, the Model layer, which lies in the bottom, usually represents and stores the data or information of the program. For many programs, this layer is the core layer that largely dominates the structure of the other two layers. Thirdly, the Controller layer, which lies in the middle, is mainly responsible for the interactions between the Model and View layers. More specifically, it accepts user instructions from View layer, retrieves data, manipulates the model, and sends back the resulting data to view for to be displayed. The above three layers are tightly connected together as well as being independent. The internal operations within each layer never affect other two layers, and each layer provides appropriate interfaces for the other layers. The essential point of this design is that the entire system could be divided into independent modules so that neither changing appearances nor changing data would alter other two layers. Hence, it greatly reduces the cost of maintaining, upgrading and testing the system.

2.3.2 RESTAPI

RESTAPI (Representational State Transfer) was proposed by Roy T. Fileding in 2000 as a simple and extensive standard for developing web applications. In fact the HTTP protocol is one typical application of the RESTAPI architecture. With the vast applications of cloud computing technology, RESTAPI is very popular used by many software architectures and developers to build large web applications. The most essential features of RESTAPI are resources, unified interface, URI and statelessness. The core concept of RESTAPI is state transfer. More specifically, the network resources and actions are clearly isolated from the state of the application. The transferring of states is described by the URI so that it is very clear for both developers and end-users. By Passing around states from back end to front end, a uniform interface is created by mapping HTTP methods to CRUD operations. The RESTAPIs, front end and back end, are stateless, so that the APIs are very efficient. Consequently, all information, including modifications, are saved and represented by the requesters, and servers contain no state information.

2.3.3 MMVC

The MMVC is an extension of MVC that optimized the Model or data layer. It is essentially the same as MVC except that it adds a view Model between Model layer and View layer. The new layer acts as an interface between Model and View layer since the data models in applications are becoming more and more complicated. We need to know three essential points about MMVC. Firstly, it is compatible with the existing MVC architecture. Secondly, it makes the software applications more testable. Thirdly, it works best with a binding mechanism. It should be mentioned that MMVC is a relatively new back-end architecture technology and is currently not used broadly in the software industry.

2.4 Conclusion

MVC and RESTAPI try to build a powerful general architecture for building various software systems. MVC tries to model the system based on actions, whereas RESTAPI mainly focuses on the data, the sate and transition to different states. MVC emphasizes dividing the internal implementations into three layers Model, View and Controller, and RESAPI decompose the web applications in terms of outside appearance, transitions of states of system. Basically, this project would try to follow RESTAPI standards for back-end design so that the programs are well structured and flexible for extension.

3 Styling framework

3.1 Overview

The styling frameworks are higher level programming languages based on CSS (Cascading Style Sheets) to make CSS (Cascading Style Sheets) development flexible and efficient. There are many popular styling frameworks for building beautiful and consistent looking web applications. We would talk about three popular ones LESS, SASS and Bootstrap.

3.2 Criteria

The core function of styling framework is to allow easier and efficient web UI design using CSS. A good styling framework will have following three features. First, it provides programming ability like logic examination, loops and functions for basic CSS code. Second, it provides rich and practical predefined templates, components, and a suite of themes for quick design of web UIs. Thirdly, it integrates other front-end languages like HTML and Javascript for flexible and easy developing.
The performance of front end styling framework is also important as it largely influences the UI design and system interaction. For instance, a good styling framework would provide rich dynamic UI components with well defined interfaces. To test whether the framework fully meets the requirements, one could compare the specs for the styling of HTML components before and after using relative frameworks. In fact, it is quite easy for testers or end-users to measure the differences between different UIs after using them.

3.3 Potential Choices

3.3.1 LESS

LESS extends CSS and introduces module conceptions into it. It provides much functionality for front-end developers beyond the basic CSS. It has similar grammar rules as basic CSS. The LESS grammar rules are similar to SASS grammar rules. Like SASS, LESS is also an extension of CSS3 that introduces rules, variables, selector and inheritance. It tries to generate well-formatted CSS codes that are easy to organize and maintain.

3.3.2 SASS

For front-end programmers of web applications, SASS provides more powerful functionality than LESS, which works more like a real programming language than LESS. However, for UI designers, LESS seems to be clearer.

3.3.3 Bootstrap

Unlike LESS and SASS, Bootstrap is one comprehensive and powerful front-end framework based on HTML, CSS and JavaScript. Generally speaking, Bootstrap tries to integrate tools like Compass, Blueprint and h5bp. Bootstrap is a comprehensive framework for web front-end development. It should be mentioned that Bootstrap use Normalize.css to reset CSS, which has becoming the practical standard (used more broadly than Eric meyer 2.0 implementation of Compass). It is also compatible with h5bp, meaning you can use h5bp and Bootstrap concurrently in one project. Generally, Bootstrap has three key features. Firstly, it includes the complete basic modules of CSS, although not as powerful as Compass. This make it possible for programmers to use basic CSS attributes for simple customization of HTML elements. Secondly, it provides some suites of predefined CSS, including one grid layout system like blueprint but with a different style. This helps programmers to quickly define the overall look of their web applications. Thirdly, it provides a group

of UI components based on jquery like dialogs, navigation menus, and edit boxes. They are all both powerful and ascetically pleasing. This may be the most powerful strength of Bootstrap. In fact, it is becoming a practical standard of most jquery-based web projects.

3.4 Conclusion

Since we are not professional UI designers and work as programmers for the AgbizClimate application We plan to use Bootstrap. Bootstrap will allow us to employ both the powerful programming ability and the rich UI components of Bootstrap without needing much graphic design.