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.