Matt Ping
Brian Wilhelm
CS 575
Final Project

Virus Information Database Solution

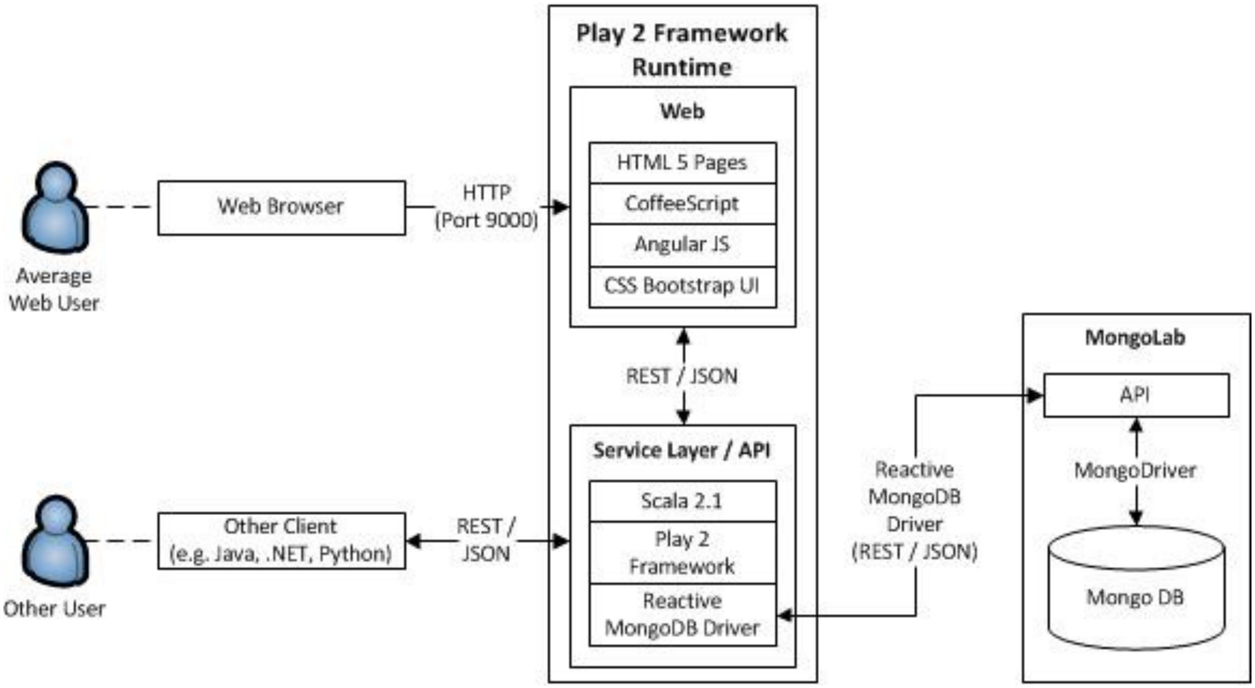
Malware has been increasing in scope and complexity in recent years. With this increase comes numerous variants of existing malware. In order to better keep track of the malware that exists, our team seeks to implement a solution for capturing information related to malware. Rather than simply providing an interface to track this information, our goal is to provide a service layer to handle the management of data. This will give users the ability to incorporate the data collected into another solution or create their own user interface if they choose.

In order to achieve the above, we selected the Play framework because of numerous advantages that it has in this context. First, in order to establish a lightweight service layer, Play provides an easy to use Scala framework for implementing RESTful services. Second, it utilizes a container-less deployment model, whereby the machine it is deployed on simply requires a JVM and the Play framework will manage HTTP requests that are received. Finally, Play also provided a number of “templates” that provide much of the structure required for our use case. For instance, the “modern-web-template” provides HTML5, AngularJS, a CoffeeScript implementation, a REST Service layer and a data access layer through MongoDB.

The combination of the Play Framework with the “modern-web-template” utilizes HTML5, CoffeeScript, AngularJS, and CSS Bootstrap UI for the front-end. This translates into a MVVM front-end that can communicate with REST based services. Communication with services is achieved through Angular JS controllers. CoffeeScript does not provide any advantages other than some improvements on the JavaScript language, but since it was included our team decided to take advantage of this feature. HTML5 and Bootstrap simply provide an easy way to construct a clean user interface.

Scala, the Play 2 Framework, and the Reactive MongoDB driver were used to create the REST layer in the application. Scala and the Play 2 Framework provided an easy to use asynchronous model through the use of Akka actors to implement externally facing REST APIs. The requests that come in then make use of a the MongoDB driver to communicate with the DB. While we could have shifted to a standard MongoDB driver, bringing in a tool that provides reactive principles, made sure that our team was utilizing reactive principles throughout the solution.

MongoDB provides a database layer utilizing BSON documents (binary JSON documents) instead of a more traditional RDBMS. A noSQL implementation provides a flexible data structure that allows for developers to more rapidly change the data structure of the project. This is useful because the solution can accommodate new or changed malware scanners, methods for identification or the inclusion of additional attributes that are relevant without major changes being necessary. To make the deployment easier for the project, our team utilized MongoLab (http://www.mongolab.com), which provides a free hosted development environment for MongoDB. Alternatively, a different Mongo deployment would be able to be used with minor modifications to the existing application.



The above diagram shows the various components of the architecture for the project. As depicted in the diagram, a user can access the solution through various HTML pages. This provides simplistic search, create, and update capabilities. These HTML pages then communicate with a REST service layer written in Scala. While the deployed pages communicate with the REST service layer, any other user who wished to write their own application could communicate with these same services as shown by the “Other User” actor. Bundled with our solution are three simple CURL commands that illustrate a connection directly to the REST APIs rather than using the HTML pages. From this service layer, the database operations are handled via a MongoDB driver.

README section

Virus Information DB

Instructions

In order to use this solution, go to http://www.github.com/CS575-Final-Project/ and clone the contents to your desktop.

Open a terminal and navigate to the “CS575-Final-Project” directory that you just cloned.

Run “./activator run” (or “activator.bat run” if on Windows).

Navigate to “http://localhost:9000/“ in the browser of your choice.

Video Link

You can view a video of this application being used at http://youtu.be/a3UQPCeoo9k

Document

It is included in the GitHub repository. Alternatively, it was also submitted as part of the BBLearn submission.

Addendum

Please note that if you wish to use your own implementation of a MongoDB, you will have to modify the following files and parameters:

File	Parameter
/CS575-Final-Project/conf/application.conf	mongodb.uri
/CS575-Final-Project/app/controllers/Viruses.scala	*def collection..(“virus_info_db”)

*Please note that this change is only necessary if you wish to have a different MongoDB collection name than “virus_info_db”.