Gabe Sequeira-Bacher

Antonio Fredericks

CS 230 – Operating Systems

Tuesday, June 7th, 2022

Journal for Module Four

In this course’s fourth week, I was tasked with developing a REST API which allowed users to interface with the server by way of a client-side browser. The established relationship in such an interface setup allows for the workload to be distributed between the front and back ends, as the two ends do not have resources in common. After a request to the server is sent, it is processed and returned to the user.

A few good things about this relationship are it allows for all of the data to be in the same place in the network, it doesn’t matter in which language you are programming in, and the client- and server-sides of the system can then scale asynchronously.

On the other hand, it does allow for multiple forms of attacks to the system: These could include Phishing, or stealing of private information at some point in the request to the server, a virus could leak into the server, or requests could be changed by malicious actors.

This project uses a REST API style to interface with the server. The framework has a few fundamental design principles that set it apart from other such frameworks as REST. I will attempt to describe them here.

* Uniform Interface: Every API request for the same resource should be the same, and each piece of data only belongs to one URI.
* Client-server decoupling: Client and server apps should be independent of each other.
* Stateless: Each request includes all the information needed in order to process it.
* Cacheability: Resources are cacheable to allow for greater speed.
* Layered system architecture: Layers are able to be seen from the client and server sides.
* And the server may return code for the client side to run.

The API holds all the information pertaining to the credentials of each user in the system. The role of the Authenticator class is to match what the user has entered in with what’s in the API. The classes are used by Dropwizard via the @AUTH annotation.

In future versions of the gameauth application, the various “roles” such as user, admin, and guest would be given permissions according to their role in using the service. This would make sure that Admins had all the permissions an admin should have and prevent users from having the same.

Sources:

<https://www.ibm.com/cloud/learn/rest-apis>

<https://codeburst.io/understanding-authentication-and-authorization-in-dropwizard-app-kotlin-e593d2052f33>

<https://www.dropwizard.io/en/latest/manual/auth.html>