Centric Software Take-Home
REST API Design Document

Michael Schroeder

Overview

I wanted to create this document as a way to understand my thought process without flooding my written code and readme full of unnecessary comments. I will use comments if it may seem necessary, but I wanted to keep those brief, and explain any design choices here in this document. I hope this document serves as a way to explain how I might tackle a future problem of this scale, as this was something I had to get used to doing for most of my undergraduate courses when designing a program of this scale.

The overview of this program is to test knowledge on REST api's and SQL databases pursuant to the role responsibilities as stated in the Centric Software job listing. This database will be able to support simple REST api calls, such as "POST and GET". Not only will the program support basic API calls, but it is designed to handle exception calls, error handling, and pagination. The pagination simply means that it will return the JSON data in a specific order, and you can sort and categorize the information by the number of pages and amount of products, per page.

Another part of the program is testing. As stated in the initial request document, tests must be considered. The type of tests will be functional testing, and unit tests. These tests will consist of checking that the information being passed into the database will conform to the requirements of the take-home document. As well as that the API calls are handling the information correctly.

Goals

My personal goals for this project are to create a program that is capable of being lightweight, fast, and built from the bottom-up to be scalable. While this may seem unnecessary, my priority is to write code that is detail oriented, using the SpringBoot micro service library. This seems to be a very common framework approach while using Java, so my goal is to make my approach simple enough to be able to achieve the goal, while learning as much as I can. While I considered using the Javalin micro framework, I had to decide between a potentially newer, and unique take on launching a RESTful API, and a well established and documented framework such as Spring. I decided to go with Spring, to err on the side of caution and be overwhelmed with information and documentation, rather than have to spend a lot of time trying to figure it out on my own. My goal ultimately is to learn as much as I can for this project.

Learning Opportunities

My biggest learning opportunity was the choice of framework used. I really wanted to use the Javalin framework instead of Spring-Boot. Spring-boot has tons of good documentation and tutorials out there to get someone like me started, however Javalin potentially looks to be the best way to write a program of this size. It is super quick to get off the ground, however, pagination is where I became stuck. There is currently no support for server-side pagination that would be as simple as calling the spring-boot JPA repository, and of course, that could significantly harm the program's ability to use less memory and scale.

Another learning opportunity was testing. Testing was never something that was truly emphasized on during my coursework, so finding the time to dig in deep here was a unique opportunity. I wished I had completed more unit tests, but I fear that this could have potentially derailed me from submitting the baseline requirements in an orderly fashion. The tests currently achieve a simple goal of making sure that the API calls are successful, and in some situations, properly checking that they are unsuccessful. Overall, this challenge was a great learning opportunity for myself.