The Asset Management Digital Challenge

Using Spring Boot (https://spring.io/guides/gs/spring-boot/), develop a working prototype which satisfies the following criteria:

A Retail Manager (using a RESTful client e.g Chrome's Postman), wanting to keep track of their shops, does a RESTful POST to /shops with a JSON of shopName, shopAddress.number and shopAddress.postCode to the Shops API (microservice) where shops are stored in memory. Permanent persistence of the data is not required.

The application can be used by multiple users at a time (no login functionality is required). Shops are identified by their unique names. If user A adds a shop that was already added by user B, the service should replace previous version and REST response to user A should contain information about the version that was replaced. If two users submit a shop at the same time, exactly one of them should get information about replacing another version of the shop.

Whenever a shop is added the service calls the Google Maps API. The Google Maps API responds with the longitude and latitude, which allows the shop data to be updates with longitude and latitude.

A customer, using their geolocation on their phone, wants to find the store that is closest to them. The Shops API will have the customer's longitude and latitude, but also the longitude and latitude of each shop to do the calculation.

The customer does a RESTful GET to the Shops API, providing their current longitude and latitude (e.g. URL request params), and gets back the address, longitude and latitude of the shop nearest to them.

Guidance

Please provide us with build/test/run (in a readme) instructions as well as the codebase, preferably via github. We currently use Gradle (http://gradle.org/) as our build runner.

We wouldn't ask anyone to do a coding puzzle that we haven't done ourselves, so the people looking at your code understand the problem we're asking to be solved. It took us around 4 - 6 hours of effort when doing this, so please use this as an indicator of effort required.

We are keen to see how much you think is enough, and how much would go into a Minimum Viable Product. As a guide, clean, elegant and simple code wins over feature rich every time

Do you test drive your code? This is something we value. We want to see that you are familiar and able to create a build process for your code that means it is easy to build, test and run. Any indicator of design (DDD, or design patterns) would make us smile, as well as BDD and TDD approaches to dev. We also pay a lot of attention to producing thread-safe code.

We also consider the extensibility of the code produced. Well factored code should be relatively easily extended. Some questions we'll probably ask:

- How you would expand this solution, given a longer development period?
- How would you go about testing this solution?
- How would you integrate this solution into an existing collection of solutions used by the Retail Manager?
- How would you go about deploying this solution to production systems?

If you have any questions about this challenge, please contact seb.leal-bennett@db.com