Technical Assessment - Backend Developer

Create a RESTful API that would allow an application to create shipping orders for products from the Warehouse to a Shop.

Context

This old fashioned company is trying to take a step towards the digital transformation. When a shop gets out of stock of some product, the manager creates a shipping order using the new application we're building. In the warehouse, this order is reviewed by a warehouse officer and when the stock is ready, the order is shipped. Finally, once the order is in the shop, the manager confirms the delivery.

Some things to keep in mind:

- A Product has a name, description, brand and stock
- A Shop has a name, city and address
- A Shipping order has three possible statuses: **pending**, **in progress** and **delivered**
- Only Warehouse officers can put an order in progress
- Only Shop managers can create orders
- Only Shop managers can mark an order as delivered

Required endpoints

- Create a new order
- Return all the products with the ability to filter them by brand
- Update the status of an order
- Return all the pending or in progress shop orders sorted by last update date.

Authentication

This API will be private, used by a mobile app or server side applications so every call to the API will include an API token to validate. Please indicate in the API documentation where in the request we should place that token.

***Note**: for simplicity make the token an environment variable.*

Technical requirements

- The solution should be clear, concise, efficient and maintainable.
- Express.js

- PostgreSQL
- Javascript ES6+
- Please do not use auto-generated code, start from scratch
- Please use a public repository on Github or Bitbucket and treat this assignment as a work project, doing frequent and descriptive commits as this will be evaluated as well.
- You are of course free to use whatever resources or references that are available to you but it is expected that the design/solution will be 100% your own.

Bonus!

The following features are not required, but it would be awesome if you could develop them.

Orders Dispatching

When the order is updated to "**in progress"**, the dispatch area must be notified in order to prepare the package and ship it. For this, the order must be sent to a message queue. A process running in the dispatch area computer will print the order & delivery details. As the communication protocol with the printer is not defined yet, for now just show the details in the console.

***Note**: the dispatch area computer does not have access to the database*

Message Queue

Since all the infrastructure will run on Amazon Web Services, for the message queue we recommend using ElasticMQ (https://github.com/softwaremill/elasticmq), an in-memory message queue system compatible with Amazon SQS. The aws-sdk package should be used to publish and consume the messages on this queue.

You need to have docker installed on your machine. In order to get it running follow the next steps:

1) Create a **custom.conf** file with content below

```
include classpath("application.conf")

queues {
 dispatch-queue {
   defaultVisibilityTimeout = 10 seconds
   delay = 5 seconds
   receiveMessageWait = 0 seconds
   fifo = false
   contentBasedDeduplication = false
 }
}
```

2) Run **`docker run -p 9324:9324 -p 9325:9325 -v \`pwd\`/custom.conf:/opt/elasticmq.conf softwaremill/elasticmq-native`**

Once the queue is running, you're ready to publish/receive messages.

- The queue URL is **http://localhost:9324/queue/dispatch-queue**
- You can go to http://localhost:9325/ to see the current system status.

Deliverables

You should send us back a Github/Bitbucket repository link, with clear instructions on how to build and run the application and the API documentation as well.

This is a simple assessment but you should treat it as an application that will have a lot of traffic, that other developers will work on it simultaneously and that will be extended with new features. You should be familiar with concepts like TDD, Modularity, OOP, etc.

As a developer sometimes we try to overthink stuff, focus on the requirements and deliver clean, readable and reusable code.