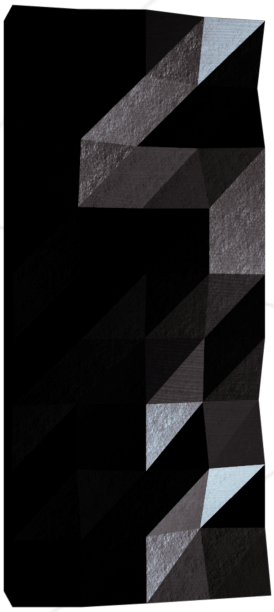

Microshop

Micronaut

Summary:
Now you will master the skills of creating microservice application and deploying them to the Cloud using Micronaut framework and various Oracle technologies.

Version: 1.00

Contents

Chapter I

General Rules

- Use this page as the only reference. Do not listen to any rumors and speculations about how to prepare your solution.
- You mus use the latest LTS version of Java. Make sure that compiler and interpreter of this version are installed on your machine.
- You must use GraalVM to run your code.
- You can use IDE to write and debug the source code (we recommend IntelliJ Idea).
- The code is read more often than written. Read carefully the document where code formatting rules are given. Make sure you follow the generally accepted Oracle standards
- Pay attention to the permissions of your files and directories.
- To be assessed, your solution must be in your GIT repository.
- Have a question? Ask your neighbor on the right. Otherwise, try with your neighbor on the left.
- Your reference manual: mates / Internet / Google. And one more thing. There's an answer to any question you may have on Stackoverflow. Learn how to ask questions correctly.
- Read the examples carefully. They may require things that are not otherwise specified in the subject.
- And may the Force be with you!
- Never leave that till tomorrow which you can do today ;)

Chapter II

Mandatory part

This project is about creating an e-commerce platform built as a set of microservices.

II.1 Overview

Thanks to your website, users will be able to buy a various set of items. You will provide a nice user interface.

Your work has to comply with the following rules:

- Your application should be Cloud Native.
- Your website backend must be written in `Java` using `Micronaut framework` as separated microservices. We will detail every microservice later in this document.
- Each microservice should be a separated `Gradle` and/or `Maven` project.
- The frontend must be written with a `TypeScript` framework of your choice (React for example).
- You are free to use any library you want to in this context. However, you must use the **latest stable version** of every library or framework used in your project.
- You must use a `Oracle` database. That's it, no other database.
- Your website must be a single-page application. The user should be able to use the `Back` and `Forward` buttons of the browser.
- Your website must be compatible with the **latest stable up-to-date version** of *Google Chrome* and one additional web browser of your choice.
- The user should encounter no unhandled errors and no warnings when browsing the website.
- Everything in a local deployment has to be launch by a single call to: `docker-compose up --build`

When your computers in clusters run under Linux, you will use Docker in rootless mode for security reasons. This comes with 2 sideways: 1) your Docker runtime files must be located in /goinfre or /sgoinfre. 2) you can't use so called "bind-mount volumes" between the host and the container if non-root UIDs are used in the container. Depending on the project, your situation and the context, several fallbacks exist: Docker in a VM, rebuild you container after your changes, craft your own docker image with root as unique UID.

II.2 Security concerns

In order to create a fully functional website, here are a few security concerns that you have to tackle:

- Any password stored in your database must be **hashed**.
- Your website must be protected against `SQL injections`.
- You must implement some kind of server-side validation for forms and any user input.

`Please make sure you use a strong password hashing algorithm`

`For obvious security reasons, any credentials, API keys, env variables etc... must be saved locally in a .env file and ignored by git. Publicly stored credentials will lead you directly to a failure of the project.`

II.3 Microservices

Your application will be divided (at least) in the following microservices:

- User : Microservice responsible for managing user accounts, registration and login.
- Catalogue : responsible for managing the information about the range of products available.
- Carts : Manages shopping carts in the shop application.
- Orders : Responsible for managing information for a particular product order including card and shipment information.
- Payment : Simulates a Microservice that would handle payment processing.
- fullfillment : Simulates processing the final order for product sold on the shop.
- Frontend : provides the HTML/JS UI of the application.
- Assets : Provides your application assets (such as images and large files).
- Api : An HTTP gateway application that is responsibile for security and routing to the different Microservices that occupy the backend and are not directly exposed over the internet

Each microservice must provide a RESTful API, each microservice will have at least the minimum endpoints to be functionnal (for example register and login for users, list of products and their categories for the catalogue, ...)

II.3.1 User

Your are free to choose the user's fields, but it should contain at least a username, a password, first name, last name, email, phone, addresses and cards.

II.3.2 Catalogue

You are free to choose the products of your catalogue. Your products should have a name, a description, a quantity, a price, an image and anything else your judge necessary.

II.4 Deployment

In addition to local deployment using Docker, you should also deploy your application the Oracle Cloud, your are free to choose how to deploy it.

Oracle Cloud Infrastructure and Cloud Native Application Deployment to Oracle Cloud

Chapter III

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your files to ensure they are correct.