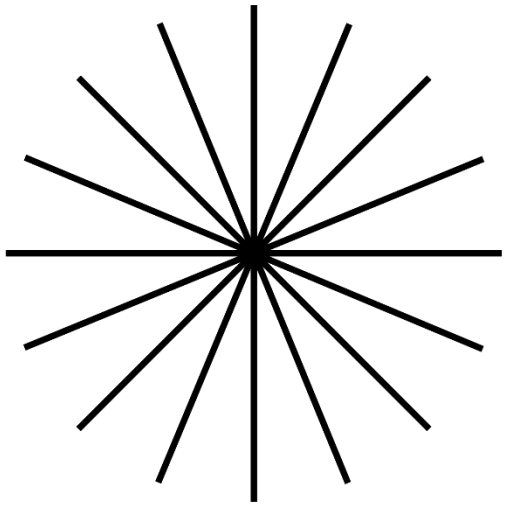
Backend Assignment

Coffee Shop

Vega IT



vega IT

The assignment

Create an app that simulates a typical coffee shop.

The app consists of two parts, frontend and backend. No strict requirements will be defined, and you should come to a design of the UI and backend architecture on your own. If you feel like it, improve the business logic and document it. It's a plus!

Coffee shop ☕

The coffee shop has 3 baristas, and one barman accepting the orders. Every barista has its own espresso machine with a limited amount of coffee. After all of the coffee is spent, the barista must refill the grinder with fresh coffee, but this takes time.

Customers that want a coffee to go, order the coffee at the bar by placing orders one by one. Orders can be one of the following:

- Espresso (35 seconds / 7 grams) - 1$
- Espresso doppio (45 seconds / 14 grams) - 2$
- Cappuccino (60 seconds / 7 grams) - 2.5$

Each type of coffee takes a fixed amount of time for the barista to complete it. Also, each type of coffee takes a certain amount of coffee in grams. When the coffee shop is opened in the morning, every coffee grinder is at its full capacity of 300 grams.

When there is not enough coffee in the grinder, the barista must refill it.

This action takes him 2 minutes. During this time, he cannot make new coffees.

Ordering

The orders can be placed in one of two ways:

- Order at the table
- Coffee to go

Order for tables is placed by customers using a tablet located at each table at the coffee shop. The tablet shows a WebUI where they can see a list of available coffees with details.

For this kind of order, customers are notified when the coffee is ready, so they can pick it up.

Coffee to go is ordered at the bar. The barman takes these orders and passes them to available baristas. If all baristas are busy (already making the coffee), the barman will accept up to 5 orders.

Requirements:

- Placing the order to the bartender (coffee to go) is done by the simple HTTP endpoint.
  - If the order is accepted, a successful HTTP response is returned (without waiting for coffee to be made)
  - If ordering is not possible, an unsuccessful HTTP response is returned.
- Placing the order using WebUI invokes another HTTP endpoint.
  - All orders from WebUI are accepted (there is no limit)
- Orders for coffee to go needs to be prioritized over orders at the table. Those are the people waiting in line so we should serve them as fast as we can. Orders at the table can last longer to brew.
- The WebUI should show a nice picture of each available coffee. People just don't know the difference between cappuccino and macchiato.
- WebUI should notify a user that his coffee is ready

Admin functionality

The owner occasionally changes the types of coffees so he must be able to perform these changes.

Requirements:

- The backend must expose endpoints to the following operations:
  - add/remove/modify coffees
- Coffee has:
  - Type
  - Price
  - Picture
  - Amount of coffee in grams
  - The time necessary to brew
- No UI is necessary for admin operations

Additional Information

- Add a readme with instructions on how to set up, run, and use the application.
- Any additional comments are more than welcome.
- Use Git for version control.
- For any 3rd party software like Database, use a docker container.
- Create a Postman collection, or similar mechanism for testing the backend side

Repository

The project should be located in a GitHub repository named
“NameSurname_Omega_CoffeeShop.

If you do not have a Github profile, create one.

After you are finished, transfer the repository ownership to a profile you received in the mail with instructions.

After this step, you will no longer see the repository.

Transferring the ownership is done by:

○ Repository home page > Settings > Options (Side menu) > Transfer ownership (at the bottom of the page)

The username for the new owner is provided in the mail.