ORACLE

Senior Backend Developer

Practical test

Objective

This test is designed to evaluate your ability to build a web api using a minimal amount of information.
The test contains two exercises, and there is a limit of 24 hours to send the results back.

Knowledge

Node.js, MySql/PostgreSQL

Guidelines

Each exercise focuses on a specific topic that a Backend Developer must master. Some exercises have bonus sub-exercises. It is recommended to first finish the test and then go after the bonuses.

ORACLE

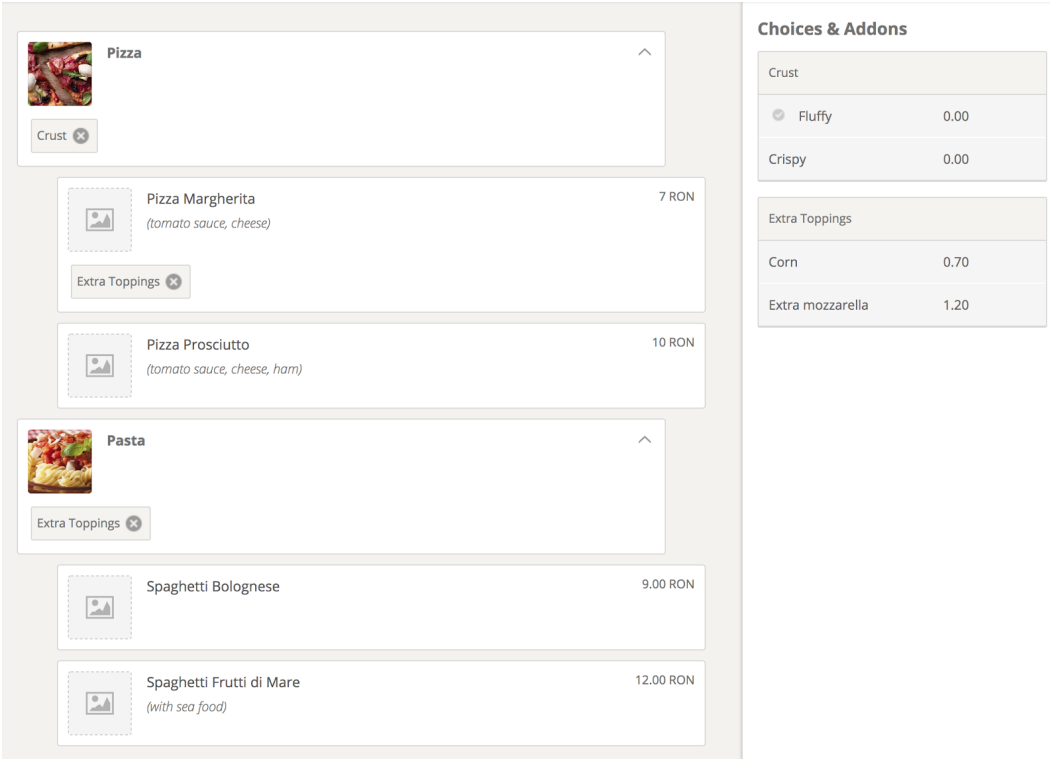
1. GET Food Menu

A different team will build a Food Menu management interface containing the following components:

- Food categories (Pizza, Pasta …)
- Food items (Pizza Marguerita, Pizza Prosciuto ...)
- Choices & Addons (Crust, Extra Toppings ...)
- Choices & Addons items (Crispy, Fluffy …)

This team has provided you with a JSON of how they expect the data to be returned by the api endpoint you are building.

Here is also a preview of the design of what they are building:



Using the information above, design and implement a DB schema able to store the menu entities (categories, items, choices, addons etc), setup a Node.js server and configure an API route which can fetch menu information from DB and returns it to the client as JSON. We suggest to use the JSON provided together with the test in order to get inspiration of how the returned payload should be.

HINTS:

- The food menu contains food menu categories, a menu category contains food menu items, etc
- Choices & Addons groups can be linked both to food categories and food menu items.

ORACLE

2. Menu management REST API

Build a REST API which enables the management of at least two of the menu entities (categories, items, choices, addons).

HINTS:

- Menu entities are very similar to each other
- Choices & Addons groups can be linked to multiple categories or items

DELIVERABLES:

Zipped folder containing

- all the code you created (javascript files)
- a dump of the database schema + the menu entities data

BONUS:

- Write unit tests (at class/module level) or api tests (at api level, using http)
- Generate (automated) documentation for the API that you just created