Exam project work

Tecnologie e Applicazioni Web, a.a 2022/2023

The goal is to develop a full-stack web application, comprising a REST-style API backend and a SPA Angular frontend to manage the orders of a restaurant. The system must handle the following kind of users:

Waiters

Each waiter carries a mobile device (smartphone or tablet) with which she/he can take orders at the tables.

Cooks

The cooks are notified in real time about the food ordered at each table by the waiters. Orders are entered in a queue and grouped by table and by production time so that the preparation of dishes for the same table can be served at the same time. A cook, using her/his user interface, can notify the system when a disk preparation begins or when is completed. When all the preparations for a given table are complete, the waiter associated with that table is notified so that she/he can proceed to serve.

Bartenders

Bartenders are conceptually like cooks, but they only deal with drinks. Bartenders receive notification of each table's orders and send notification to the waiters when the tray of drinks of a specific table is ready for service.

Cashier

The cashier can produce a receipt for a table with the bill to be paid. The cashier also has a view of the statistics on the status of the orders, in particular:

- Which tables have orders in preparation
- Which waiter is associated with each table
- Which tables are free/occupied
- How long are the kitchen preparation queues

Finally, the cashier can compute the total profit for a day.

Architecture

The system must comprise:

- A REST-style **backend** webservice, implemented in TypeScript and running on Node.js. The service must use:
    - MongoDB DBMS for data persistence
    - Express.js for routing

- A web **front-end** implemented as a Single Page Application (SPA) using the Angular framework.

Optional (suggested) requirement: each component (Backend, MongoDB, Front-end) should run in a separate Docker container.

Requirements

Your application must (at least) implement the following features:

1. User management
    a. Registration of new users with the following roles: waiters, cooks, bartenders, and cashier. The cashier also acts as an administrator;
    b. The administrator can delete existing users;
    c. User login. **All the system functionalities are accessible only after a mandatory login**.
2. Orders and tables management (waitress only)
    a. Visualization of “free” and “occupied” tables;
    b. Change the status of a table from “free” to “occupied” by a certain number of customers. The number of customers must be less than or equal the available seats at the table;
    c. Adding of one or more dish orders for a particular table. Cooks are notified when the ordering phase of a table is completed;
    d. Adding of one or more drink orders for a particular table. Bartenders are notified when the ordering phase of a table is completed;
    e. Notification to the waiter assigned to a table when dishes and/or drinks are ready to be served to that table.
3. Management of the food preparation queue (cooks only):
    a. Real-time visualization of the food preparation queue. Assume for simplicity that there is a single preparation queue for all the tables, and each table is served with a FIFO logic.
4. Management of the drink preparation queue (bartenders only):
    a. Real-time display of the drink preparation queue for each table. When the tray of drinks is ready, the bartender notifies this event to the system which will forward it to the waiter assigned to that table (like what happens for cooks).
5. Cash management (cashiers only):

a. Computation of the bill associated to a table and creation of the receipt with the list of orders consumed. When the receipt is produced, table status switch from occupied to free and can be reused by the waiters:
b. Visualization of free and occupied tables:
c. Visualization of the orders placed and awaiting preparation for each table;
d. Display of statistics on each waiter and cook, like the number of customers served, dishes prepared, etc..

You are welcome to creatively enrich the project with additional features. This does not imply to get honors (Lode) but concur to define a positive grade.

Submission

Projects must be submitted via Moodle at the following page:
https://moodle.unive.it/mod/assign/view.php?id=650404

If working on a group, each group member must submit the same file named: `<group_name>.zip`.

The package must contain:

- **The report of each student**, in PDF format, named `<student_firstname>_<student_surname>_<matriculation_number>.pdf`
- A `README.txt` file with instructions on how to run the entire application. For example, you can briefly summarize the shell commands to compile the sources, start backend and frontend, etc.
- All the application source code, possibly divided in backend and frontend.

NOTES:

- Do not submit any external library. In other words, remember to delete all the `node_modules` directories before submission.
- Students are responsible for group creation. Moodle will not help nor enforce group submission. Individual students can simply use their surname as group name.

Report

Together with the application, you must submit a report describing the system architecture, the software components used the way in which they contribute to achieving the required functionalities. The report is **strictly individual** and will serve as a basis for the oral discussion of the project.

The report must contain:

- A description of the system **architecture**, listing all the different components and their relationship (i.e. How they work together to implement the application requirements).
- A description of the **data model**, including the collections and the structure of the documents inside each collection.
- A precise description of the **REST APIs**. Such description must contain the list of endpoints together with their parameters and what data is exchanged (give examples in JSON format whenever possible).
- A description of how the user **authentication** is managed, with the associated workflow.
- A description of the Angular frontend, with a concise list of the implemented **components, services, and routes**.
- Some examples, possibly with **screenshots**, showing the typical application workflow for each different user role.

Q&A

For any question about the project work (or the course in general) don't hesitate to contact the teacher at:
filippo.bergamasco@unive.it

I'll be happy to arrange a meeting if needed!

Additional information about the exam is available at:
https://moodle.unive.it/mod/page/view.php?id=614812

Prof. Filippo Bergamasco,
Tecnologie e Applicazioni Web, a.a. 2022/2023