VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

PROGRAMŲ SISTEMŲ KATEDRA

**Quick Fix**

**Lab 2**

**Technical Design Document**

    Darbo autoriai:

Kristupas Norvaiša

Kamilė Čiulčinskytė

Rokas Gervetauskas

Matas Bobin

Vilnius, 2022

**Contents**

[Introduction 2](#_Toc1679585979)

[General comments 3](#_Toc1617268807)

[Comments on Data model/Schemas in SWAGGER 3](#_Toc2103481863)

[Instructions on how to view SWAGGER yaml file 4](#_Toc1963983277)

[Clients 5](#_Toc1516960321)

[Scalability 5](#_Toc2058983350)

[Deployment diagram 6](#_Toc1500235505)

# **Introduction**

This technical design document reviews the PRADa team’s design for the Point of Sale system. Also it contains instructions on how to comfortably view SWAGGER *yaml* file that contains system endpoints description.

# **General comments**

* Design lacks distinct versatility
* No services implemented
* A lot of necessary user stories are missing

# **Comments on Data model/Schemas in SWAGGER**

Ingredient:

* Ingredient‘s OutOfStock bool was changed into a StockStatus with possible enum values: [InStock, ComingSoon, OutOfStock] in order to fulfill the need to specify, whether a product has been ordered and is on the way to the inventory,
* „As a manager I want to order selected items, so that the bar would not have anything out of stock“ user story is out of scope, instead ingredient‘s status can be updated from OutOfStock to ComingSoon,
* Neither of the mentioned actors could add an ingredient entry to the inventory.
* "As a worker I want to contact the bar manager, so that he knows which products are missing." - there is no need to contact the bar manager because now worker can edit status of the ingredient (InStock, ComingSoon, OutOfStock).

Order:

* There was no clear tracking of an order‘s status, Status field with possible enum values was added: [Placed, Approved, Completed, Cancelled]
* Fields OrderDate and CompletionDate were added.
* Added Customer’s ID who initially placed the order
* Added payments array . If bill was spllitted there are more than one payments

Reservation:

* There was no tracking of a reservation‘s status, Status field with possible enum values was added: [Placed, Approved, Completed, Cancelled]

Report:

* It wasn‘t mentioned what‘s supposed to be included in a periodical analytics data, therefore it contains TotalRevenue obtained during a specific period of time.
* „As a manager I want to send analytics to the bar owner, so that he could see the bar’s performance “ – doesn‘t seem like it should belong to the system‘s scope

User

* UserType attribute type wasn’t specified. Added Enum type (possible values: staff, customer)

StaffMember

* Salary type changed from integer to double

Business

* Added ManagerID attribute. It’s an ID of a given business manager
* Location changed from string to Address object array. It’s holding a array of locations that belong to the same business

Address

* Added “Address” table that holds the address objects of businesses

Taxes

* Array of payments now holds a payment objects instead of double values

Payment

* “Discount” format changed to percentage
* Added CustomerID which is the ID of a user that made a payment
* Added OrderID which is the ID of a order that payment was for
* Added PaymentOptions enum (possible values: Cash, Check, Credit card, Debit card, Online payment)
* Added Status enum (possible values: Completed, Cancelled, In progress, Waiting)
* Added TimeWhenCompleted (format: date-time)

# **Instructions on how to view SWAGGER yaml file**

1. Go to <https://editor.swagger.io/>
2. Copy-paste contents of the attached file (*API\_endpoints.yaml*) to the editor on the left side of the screen.
3. Visualization of the endpoint descriptions should be visible on the right side of the screen.

# **Clients**

This system is for the residents of Lithuania.

Our clients:

* Teenagers (as young as 14)
* Adolescents
* College students
* Adults
* Business owners
* Elderly (up until year 80)

Regarding Lithuania's demography as of 2022 our possible maximum client number should reach around 2.2 million.

We predict that client sessions would consist of a more or less equal amount of writing and reading activities.

# **Scalability**

Coding aspect:

To be able to improve scalability of this application, we will try to incorporate load tests into our CI/CD processes. That way we will make sure that our application will remain scalable, as well as, the application will be free from “bottlenecks”.

Database aspect:

In addition, to make sure that the database will stay efficient under heavy loads, we will try to implement indexing strategies and optimize SQL queries, so that searching, sorting, filtering (etc.) of various materials would slow down the application as much as possible. To optimize our database we would use indiscriminate split technique (spliting by categories, names, etc.). To further optimize application functionality all most popular “GET” requests will be cacheable.

Hardware aspect:

To maximize scalability of the application, we would have at least one server for a given region that the application is available in. Moreover, if number of users will rapidly grow in given region we will additional servers to given group. “Proxy” servers will assure security of the group and “Worker” servers will respond to the client requests. “Dispatcher” server will forward requests to the nearest server group. “Load Balancer” will distribute load between local servers that way avoiding any delays.

# **Deployment diagram**

Graphical user interface, diagram

Description automatically generated