Proyect: Restauran Order API
Author: Maicol Garzon Melo

1. Functional Requirements

1.1. Order Management

1.1.1. The system shall allow a waiter to create a new order specifying:
- 1.1.1.1. One (1) base
- 1.1.1.2. Up to three (3) proteins
- 1.1.1.3. Up to five (5) toppings
- 1.1.1.4. One (1) drink
- 1.1.1.5. The table number

1.1.2. The system shall validate that the maximum of 3 proteins and 5 toppings per order is not exceeded.

1.1.3. The system shall allow retrieval of an order by:
- 1.1.3.1. Order ID
- 1.1.3.2. Table number (returning all orders associated with that table)

1.1.4. The system shall allow deletion or cancellation of an order:
- 1.1.4.1. By order ID (single order)
- 1.1.4.2. By table number (all current orders associated with that table)

1.2. Table Association

1.2.1. The system shall support multiple orders for a single table.

1.3. Order Status

1.3.1. Each order shall have a status field with possible values: pending, preparing, served, cancelled.

1.3.2. The system shall allow status updates for each order.

1.4. Pricing

1.4.1. The system shall return a total calculated price per order based on the selected base, proteins, toppings and drink.

1.4.2. Prices shall be associated with each item (base, protein, topping, drink) in the system.

2. Non-Functional Requirements

2.1. Performance

2.1.1. The system shall respond to any valid API request within 1 second under normal load.

2.1.2. The system shall support at least 10 concurrent API requests without performance degradation.

2.2. Scalability

2.2.1. The system shall be designed in a modular way to support future extensions such as:
- 2.2.1.1. Custom pricing
- 2.2.1.2. User authentication (e.g. waiters)
- 2.2.1.3. Multi-branch support

2.3. Availability

2.3.1. The system shall maintain at least 99% uptime during business hours

2.4. Maintainability

2.4.1. The code base shall follow a modular structure separating routes, business logic and database access.

2.4.2. Code shall follow consistent naming conventions and include inline documentation for key functions and decisions.

2.5. Data Integrity

2.5.1. All business rules (e.g. max item limits) shall be enforced both at the application level and in the database via constraints.

2.5.2. Orders violating rules shall be rejected with clear, user-friendly error messages.

2.6. Security

2.6.1. Inputs shall be sanitized to prevent injection and basic attack vectors.

2.6.2. In future iterations, endpoints will be protected to allow only authorized users or systems.

2.7. Portability

2.7.1. The system shall be containerized using Docker for easier deployment and development.