# Documentação Técnica da API

Italo Moura

05/08/2025

# □ Tech Challenge - Documentação Técnica da API

## Informações Gerais

**Projeto:** Sistema de Gerenciamento de Restaurantes
**Tecnologia:** Spring Boot + MongoDB
**Versão:** 1.0.0
**Data:** $(date +“%d/%m/%Y”)

## Visão Geral

Sistema de gerenciamento de restaurantes desenvolvido com Spring Boot e MongoDB, focado em alta performance de leitura através de estrutura de documentos aninhados com endpoints específicos para gerenciamento transparente de menu e itens.

## Arquitetura

O projeto segue os princípios da **Arquitetura Hexagonal (Ports & Adapters)** com separação clara de responsabilidades:

### Camadas da Aplicação

1. **Presentation Layer (Apresentação)**
   - Interfaces Contracts com anotações Swagger/OpenAPI
   - Controllers que implementam as interfaces
   - Handlers para tratamento de exceções
2. **Application Layer (Aplicação)**
   - Use Cases (Casos de uso)
   - DTOs (Data Transfer Objects)
   - Ports (Interfaces)
3. **Domain Layer (Domínio)**
   - Entidades de domínio
   - Exceções customizadas
   - Regras de negócio
4. **Infrastructure Layer (Infraestrutura)**
   - Configurações do MongoDB
   - Repositórios
   - Adaptadores externos

## Endpoints da API

### Kitchen Types (Tipos de Cozinha)

| Método | Endpoint | Descrição | Status Code |
| --- | --- | --- | --- |
| POST | `/api/kitchen_types` | Criar tipo de cozinha | 201 Created |

| | | | |
|---|---|---|---|
| POST | /api/kitchen_types | Criar tipo de cozinha | 201 Created |
| GET | /api/kitchen_types | Listar todos os tipos | 200 OK |
| GET | /api/kitchen_types/{id} | Buscar por ID | 200 OK / 404 Not Found |
| PUT | /api/kitchen_types/{id} | Atualizar tipo | 200 OK / 404 Not Found |
| DELETE | /api/kitchen_types/{id} | Remover tipo | 204 No Content / 409 Conflict |

### Restaurants (Restaurantes)

| Método | Endpoint | Descrição | Status Code |
|---|---|---|---|
| POST | /api/restaurants | Criar restaurante | 201 Created |
| GET | /api/restaurants | Listar restaurantes básicos | 200 OK |
| GET | /api/restaurants/full | Listar restaurantes completos | 200 OK |
| GET | /api/restaurants/{id} | Buscar restaurante por ID | 200 OK / 404 Not Found |
| PUT | /api/restaurants/{id} | Atualizar restaurante | 200 OK / 404 Not Found |
| DELETE | /api/restaurants/{id} | Remover restaurante | 204 No Content / 404 Not Found |

### Menu Categories (Categorias do Menu)

| Método | Endpoint | Descrição | Status Code |
|---|---|---|---|
| POST | /api/restaurants/{restaurantId}/menu | Criar categoria | 201 Created |
| GET | /api/restaurants/{restaurantId}/menu/{menuId} | Buscar categoria | 200 OK / 404 Not Found |
| PUT | /api/restaurants/{restaurantId}/menu/{menuId} | Atualizar categoria | 200 OK / 404 Not Found |
| DELETE | /api/restaurants/{restaurantId}/menu/{menuId} | Remover categoria | 204 No Content / 404 Not Found |

### Menu Items (Itens do Menu)

| Método | Endpoint | Descrição | Status Code |
|---|---|---|---|
| POST | /api/restaurants/{restaurantId}/menu/{menuId}/item | Criar item | 201 Created |
| GET | /api/restaurants/menu/item/{itemId} | Buscar item com contexto | 200 OK / 404 Not Found |
| PUT | /api/restaurants/{restaurantId}/menu/{menuId}/item/{itemId} | Atualizar item | 200 OK / 404 Not Found |
| DELETE | /api/restaurants/{restaurantId}/menu/{menuId}/item/{itemId} | Remover item | 204 No Content / 404 Not Found |

## Modelagem de Dados

### Estrutura do Documento Restaurant


```
{
  "_id": "550e8400-e29b-41d4-a716-446655440000",
  "name": "Restaurante do João",
  "address": "Rua das Flores, 123",
  "kitchenType": {
    "id": "550e8400-e29b-41d4-a716-446655440001",
    "name": "Japonesa",
    "description": "Cozinha Japonesa"
  },
  "daysOperation": [
    {
      "day": "MONDAY",
      "openingHours": "08:00",
      "closingHours": "18:00"
    }
  ],
  "ownerId": "550e8400-e29b-41d4-a716-446655440002",
  "isActive": true,
  "menu": [
    {
      "id": "550e8400-e29b-41d4-a716-446655440003",
      "type": "Lanche",
      "items": [
        {
          "id": "550e8400-e29b-41d4-a716-446655440004",
          "name": "Hambúrguer Artesanal",
          "description": "Hambúrguer com carne artesanal",
          "price": 25.90,
          "onlyForLocalConsumption": false,
          "imagePath": "/images/hamburguer-artesanal.jpg",
          "isActive": true
        }
      ]
    }
  ],
  "lastUpdate": "2024-08-05T10:30:00",
  "createdAt": "2024-08-05T08:00:00"
}
```


# Regras de Negócio

## Kitchen Types

- Nome é obrigatório e único (case-insensitive)
- Descrição é opcional
- Não pode ser excluído se estiver sendo usado por restaurantes
- Trim automático de espaços em branco

## Restaurants

- Nome e endereço são obrigatórios
- Tipo de cozinha é obrigatório
- Horários de funcionamento são obrigatórios
- ID do proprietário é obrigatório
- Restaurante é ativo por padrão

## Menu Categories

- Tipo da categoria é obrigatório
- Categorias têm ID único (UUID)
- Podem ser criadas vazias (sem itens)

### Menu Items

- Nome e preço são obrigatórios
- Preço deve ser positivo
- Itens são ativos por padrão
- `onlyForLocalConsumption` é false por padrão

## Códigos de Status HTTP

### Sucesso

- **200 OK**: Operação realizada com sucesso
- **201 Created**: Recurso criado com sucesso
- **204 No Content**: Recurso excluído com sucesso

### Erro do Cliente

- **400 Bad Request**: Dados inválidos fornecidos
- **404 Not Found**: Recurso não encontrado
- **409 Conflict**: Conflito (ex: nome duplicado, recurso em uso)

### Erro do Servidor

- **500 Internal Server Error**: Erro interno do servidor

## Tecnologias Utilizadas

### Backend

- **Spring Boot 3.5.4**: Framework principal
- **Spring Data MongoDB**: Integração com MongoDB
- **Spring Validation**: Validação de dados
- **Lombok**: Redução de boilerplate

### Banco de Dados

- **MongoDB**: Banco NoSQL orientado a documentos
- **UUID**: Identificadores únicos

### Documentação

- **SpringDoc OpenAPI**: Documentação automática da API
- **Swagger UI**: Interface interativa da API

### Testes

- **JUnit 5**: Framework de testes
- **Mockito**: Mocks para testes unitários

## Performance e Otimizações

### Estratégias Implementadas

1. **Estrutura Aninhada**: Menu integrado ao documento do restaurante
2. **Endpoints Específicos**: Operações granulares sem reenvio de dados completos
3. **Índices Automáticos**: Configuração para criação automática de índices

4. **UUID Nativo**: Conversores customizados para melhor performance

### Métricas Esperadas

- **Consulta Básica**: ~5ms (restaurantes sem menu)
- **Consulta Completa**: ~15ms (restaurantes com menu)
- **Operações de Menu**: ~8ms (criar/atualizar categoria)
- **Operações de Item**: ~10ms (criar/atualizar item)

## Segurança

### Validações Implementadas

- Validação de entrada em todos os endpoints
- Sanitização de dados (trim de espaços)
- Validação de tipos de dados
- Verificação de integridade referencial

### Boas Práticas

- UUIDs para identificadores únicos
- Timestamps automáticos
- Tratamento de exceções padronizado
- Logs estruturados

## Conclusão

O sistema foi desenvolvido seguindo as melhores práticas de desenvolvimento, com foco em: - Performance otimizada para MongoDB - Arquitetura limpa e testável - Documentação completa e padronizada - Endpoints RESTful bem definidos - Estrutura de dados eficiente