**Sistema de Gestão de Pedidos**

**Sumário**

1. [Introdução](https://chatgpt.com/c/3f2b27fb-ac95-4f98-bbf8-185f2f108887#introdu%C3%A7%C3%A3o)
2. [Visão Geral do Sistema](https://chatgpt.com/c/3f2b27fb-ac95-4f98-bbf8-185f2f108887#vis%C3%A3o-geral-do-sistema)
3. [Arquitetura](https://chatgpt.com/c/3f2b27fb-ac95-4f98-bbf8-185f2f108887#arquitetura)
4. [Funcionalidades Principais](https://chatgpt.com/c/3f2b27fb-ac95-4f98-bbf8-185f2f108887#funcionalidades-principais)
5. [Design dos Serviços](https://chatgpt.com/c/3f2b27fb-ac95-4f98-bbf8-185f2f108887#design-dos-servi%C3%A7os)
6. [Principais Endpoints](https://chatgpt.com/c/3f2b27fb-ac95-4f98-bbf8-185f2f108887#principais-endpoints)
7. [Documentação da API com Swagger](https://chatgpt.com/c/3f2b27fb-ac95-4f98-bbf8-185f2f108887#documenta%C3%A7%C3%A3o-da-api-com-swagger)

**Introdução**

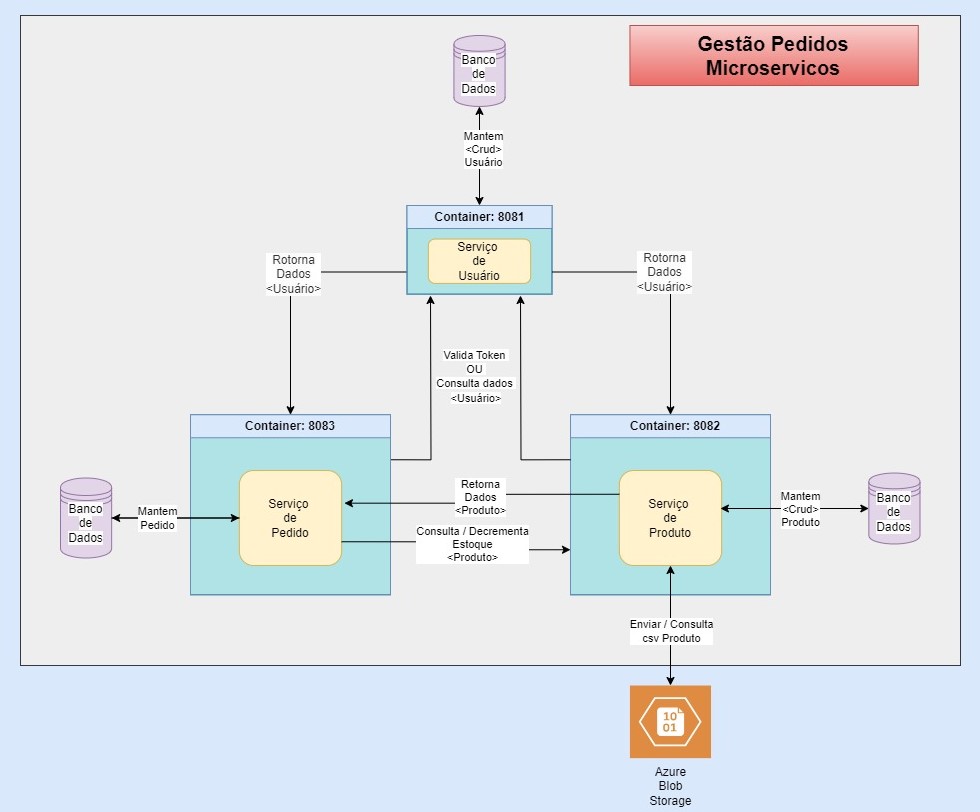
O sistema de gerenciamento de pedidos é altamente eficiente, explorando profundamente a arquitetura de microsserviços utilizando o ecossistema Spring. Este sistema abrange desde a gestão de clientes e produtos até o processamento e entrega de pedidos, enfatizando a autonomia dos serviços, comunicação eficaz, e persistência de dados isolada.

**Visão Geral do Sistema**

O objetivo do sistema é organizar eficientemente o estoque dos produtos e calcular a melhor rota de entrega, agrupando os pedidos de regiões próximas.

**Arquitetura**

A arquitetura do projeto é baseada em microsserviços, garantindo a autonomia e a comunicação eficiente entre os componentes. Utilizamos o ecossistema Spring para a implementação dos serviços e cada serviço possui uma base de dados isolada.



Componentes Principais:

* **Backend**: Baseado em microsserviços com Spring Boot.
* **Banco de Dados**: Utilizamos MYSQL para persistência de dados.

**Funcionalidades Principais**

1. **Registro de Cliente**
   * Operações CRUD para gerenciar clientes.
   * Dados básicos e endereço de entrega do cliente.
2. **Registro de Produtos**
   * Gerenciamento do catálogo de produtos.
   * Controle de estoque.
3. **Gestão de Pedidos**
   * Processamento de pedidos desde a criação até a conclusão.
   * Estados do pedido: aguardando pagamento, pago, aguardando entrega e entregue.

**Design dos Serviços**

Os serviços foram desenhados seguindo princípios de arquitetura de microsserviços, com comunicação via HTTP/REST. Cada serviço é responsável por um domínio específico e comunica-se com outros serviços conforme necessário.

Serviços:

* **ClienteService**: Gerencia operações relacionadas aos clientes.
* **ProdutoService**: Gerencia o catálogo de produtos e controle de estoque.
* **PedidoService**: Centraliza o processamento e gerenciamento de pedidos.

**Principais Endpoints**

**ClienteService**

* **POST /clientes**
  + **Descrição**: Cria um novo cliente.
  + **Parâmetros**: Dados do cliente.
  + **Resposta**: Cliente criado.

**ProdutoService**

* **GET /produtos**
  + **Descrição**: Lista todos os produtos.
  + **Parâmetros**: Nenhum.
  + **Resposta**: JSON com lista de produtos.
* **POST /produtos**
  + **Descrição**: Cria um novo produto.
  + **Parâmetros**: Dados do produto.
  + **Resposta**: Produto criado.

**PedidoService**

* **GET /pedidos**
  + **Descrição**: Lista todos os pedidos.
  + **Parâmetros**: Nenhum.
  + **Resposta**: JSON com lista de pedidos.
* **POST /pedidos**
  + **Descrição**: Cria um novo pedido.
  + **Parâmetros**: Dados do pedido.
  + **Resposta**: Pedido criado.

**Documentação da API com Swagger**

Para acessar a documentação detalhada dos endpoints das APIs, utilize os seguintes links:

* **Cliente**: <http://localhost:8081/swagger-ui/index.html>
* **Produto**: <http://localhost:8082/swagger-ui/index.html>
* **Pedidos**: <http://localhost:8083/swagger-ui/index.html>
* **Logística**: <http://localhost:8084/swagger-ui/index.html>