

# FoodDelivery - Sistema de Pedidos

Arthur Maia, Bruno Moreira Lima, Claudio Pales Costa, Gabriel Honorato  
Santos Ferraz, Kéven Patricio

<sup>1</sup>Centro Universitário de Excelência de Vitória da Conquista (UNEX)

<sup>2</sup>Sistema de informação

<sup>3</sup>arturcoqueiro018@gmail.com, bru.no@outlook.com.br, claudio.palles.costa@gmail.com,  
ferrazcoelhoaldrigo@gmail.com, patriciokeven4@gmail.com

**Resumo.** *O FoodDelivery é um projeto para ajudar restaurantes, bares, mercados e farmácias, com o objetivo de auxiliar no gerenciamento das empresas responsáveis por esses estabelecimentos. Esse objeto utiliza Java, com Programação Orientada a Objetos, esse projeto permitirá cadastrar itens com nome e preço, listar esses itens permitindo visualizar o cardápio com os nomes e valores, gerenciar clientes cadastrando seus nomes e outras informações, listar os clientes cadastrados, registrar pedidos, atualizar status, consultar pedidos por status e gerar relatórios, tanto simples como detalhados.*

## 1. Introdução

O setor de serviços gastronômicos, que engloba restaurantes, bares, mercados e farmácias, exige cada vez mais agilidade e organização para otimizar a experiência do cliente e a gestão interna. A crescente demanda por soluções digitais que simplifiquem o fluxo de pedidos impulsionou o desenvolvimento de sistemas robustos e eficientes. Neste contexto, apresentamos o FoodDelivery, um protótipo de sistema de gerenciamento de pedidos desenvolvido em Java, com interface de linha de comando (CLI).

O objetivo principal deste trabalho é demonstrar a aplicação dos pilares da Programação Orientada a Objetos (POO) na construção de uma plataforma flexível e escalável, capaz de lidar com o ciclo de vida completo de um pedido — desde o cadastro inicial até a sua entrega final. Embora seja um sistema CLI, sua arquitetura foi projetada para servir como a base sólida para futuras implementações, como a criação de APIs e a integração com bancos de dados.

O sistema foi arquitetado para gerenciar o cardápio, clientes e pedidos de forma integrada, oferecendo ainda funcionalidades de relatórios simplificados e detalhados. A sua implementação visa não apenas resolver um problema prático do setor, mas também servir como um estudo de caso sobre a importância do uso de padrões de projeto e conceitos de POO para o desenvolvimento de software robusto e de fácil manutenção..

## 2. Motivação e Contribuição

A motivação para a criação do sistema se deve à constante evolução da tecnologia e à necessidade de digitalizar os estabelecimentos. Muitos restaurantes, bares e mercados ainda fazem o gerenciamento de maneira manual ou utilizando planilhas no Excel. O nosso projeto visa otimizar esse tempo, aumentando a produtividade e reduzindo erros, pois o

sistema, ao controlar cardápio, preços e clientes em um único lugar, evita erros básicos que podem causar confusões no futuro. Além disso, o sistema é altamente escalável e pode ser evoluído para um site completo, um aplicativo para celular, integrado a banco de dados e outras tecnologias.

### **3. Trabalhos Relacionados**

O mercado de automação comercial e sistemas de gerenciamento de pedidos é vasto, com soluções que variam de plataformas de grande escala, como iFood e Uber Eats, a sistemas de PDV (Ponto de Venda) para negócios locais. Na academia, existem estudos de caso que exploram a aplicação de tecnologias para otimizar fluxos de trabalho, como o desenvolvimento de aplicativos para gerenciamento de pedidos com impressão térmica ou sistemas específicos para nichos de mercado.

No entanto, a maioria dessas soluções, tanto comerciais quanto acadêmicas, tende a apresentar uma arquitetura "caixa preta" ou a focar em funcionalidades muito específicas, limitando o aprendizado sobre o design e a flexibilidade do software.

Em contraste, o FoodDelivery se diferencia por sua arquitetura orientada a objetos modular e didática, projetada para servir como um estudo de caso prático da aplicação de conceitos de Programação Orientada a Objetos (POO). Ele não apenas resolve o problema de gerenciamento de pedidos, mas também oferece uma base transparente e de fácil expansão. A arquitetura da solução busca superar as limitações de acoplamento excessivo de outros sistemas, utilizando padrões de projeto como o Singleton para centralizar dados e a Agregação para gerenciar os relacionamentos entre classes, o que facilita futuros desenvolvimentos com tecnologias como Spring Boot e bancos de dados.

#### **3.1. Limitações**

O nosso sistema se diferencia da concorrência por ter uma arquitetura POO, pois o nosso projeto não é apenas funcional, mas também é um bom exercício para treinar POO. Além de que ele possui uma interface que pode ser executada por linha de comando, o que permite outros desenvolvedores conseguirem utilizá-lo com uma maior facilidade, além de permitir a criação de coisas maiores como apps mobile, por exemplo. Além disso, ele não foca só em restaurantes, diferente de vários outros softwares. O FoodDelivery é altamente flexível, podendo atender bares, mercados, farmácias e muito mais. Isso amplia o alcance dele e resolve problemas em vários setores, coisa que outros sistemas não fazem.

### **4. Arquitetura**

A arquitetura desse projeto é dividido em 5 partes:

#### **4.1. A pasta model**

Que contem as entidade principais:

Cliente.java - representa os clientes do sistema.

ItemCardapio.java - representa os itens do cardápio.

ItemPedido.java - representa a relação entre pedido e item.

Pedido.java - representa o pedido, seu status e os itens associados.

## 4.2. A pasta service

Onde se encontra as classes que implementam as regras de negócio e gerenciar o sistema:

CardapioService.java - gerencia itens do cardápio.

ClienteService.java - gerencia clientes.

PedidoService.java - gerencia pedidos, incluindo atualização de status e relatórios.

## 4.3. A pasta util

Fornece algumas funções adicionais

ConfiguraTerminal.java - configurações de exibição no terminal.

Input.java - trata entradas do usuário.

Validador.java - faz validações.

## 4.4. A pasta view

Responsavel por controlar a interação da interface com o usuário

MenuCardapio.java → menu de operações do cardápio.

MenuCliente.java → menu de operações de cliente.

TelaInicial.java → tela principal do sistema.

## 4.5. A pasta main

Ponto de entrada do sistema, Inicializa menus e conecta as camadas.

## 5. Resultados e Discussão



Figure 1. Menu de acesso principal da interface de linha de comando.

```

=== REGISTRAR NOVO PEDIDO ===

Clientes disponíveis:
ID: 1 | Nome: Gabriel | Telefone: 77900000000

Informe o ID do cliente: 1

Pedido #2 criado para Gabriel

--- CARDÁPIO ---
ID: 1 | pizza | R$ 50.00

Informe o ID do item (0 para finalizar): 1
Quantidade: 4
Item adicionado ao pedido com sucesso!

```

Figure 2. Fluxo de registro de um novo pedido.

```

=== RELATÓRIO DETALHADO - 30/08/2025 ===

????????????????????????????????????????????????????????????
Pedido #1 - 30/08/2025 20:36
Cliente: Gabriel (ID: 1)
Telefone: 77900000000
Status: ACEITO

Itens:
? pizza | Qtd: 3 | Preço Unit: R$ 50.00 | Subtotal: R$ 150.00

TOTAL DO PEDIDO: R$ 150.00

????????????????????????????????????????????????????????????
RESUMO GERAL:
Total de Pedidos: 1
Valor Total Arrecadado: R$ 150.00

```

Figure 3. relatório detalhado

O sistema FoodDelivery demonstrou a viabilidade de um modelo de gerenciamento de pedidos com foco em flexibilidade e organização. A interface de linha de comando (CLI) provou ser eficaz para validar a lógica de negócio principal. A Figura 1 ilustra o menu de acesso que direciona o usuário para as funcionalidades do sistema.

As funcionalidades implementadas foram organizadas em módulos claros, refletindo a arquitetura orientada a objetos do projeto. O gerenciamento de entidades como ItemCardapio e Cliente foi centralizado na classe Singleton, garantindo consistência. O ciclo de vida de um pedido foi modelado com um Enum para status, e a classe Pedido utiliza o conceito de agregação, como ilustrado no fluxo de registro na CLI (Figura 2).

A capacidade de gerar relatórios simplificados e detalhados demonstra a flexibilidade do sistema para extrair informações cruciais para a gestão. A escolha da arquitetura POO se mostrou fundamental, criando um sistema de baixo acoplamento que facilita a manutenção e a expansão. Essa estrutura permite uma futura adaptação para interfaces gráficas ou APIs REST, sem a necessidade de reescrever a lógica central.

## 6. Considerações finais

Nosso projeto visa modernizar estabelecimentos de maneira simples, prática e direta, utilizando Java e POO. Assim, além de ajudar as empresas desses estabelecimentos, ele possui algumas limitações, como não possuir um banco de dados e também não ter recursos tão avançados. Além disso, ele ainda é um protótipo e tem espaço para ser utilizado por projetos futuros, nos quais podemos trabalhar, como integrar com um banco de dados, possivelmente o MySQL, criar APIs com Spring Boot, implementar Backend for

Frontend e um sistema de notificação.

## References

Consumer Sistemas. Sistema para Restaurantes, Bares, Lanchonetes e Delivery. Disponível em: <https://consumer.com.br/> (Acesso em: 28 de ago. 2025).

MarketUP. Plataforma de Gestão e Vendas Gratuita para PMEs. Disponível em: <https://marketup.com/> (Acesso em: 28 de ago. 2025).

PONTES, I. M. T.; CAMPELO, J. S. P. Aplicativo gerenciador de pedidos de restaurante com conexão para impressora térmica. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação, UFCG).

NUNES, J. M. AskLunch: Aplicativo para pedidos e gerenciamento de marmittas. Trabalho de Conclusão de Curso (Bacharelado em Análise e Desenvolvimento de Sistemas, IFG Câmpus Jataí).

SILVA, E. D. Uma arquitetura de software para sistemas de gerenciamento baseados em IoT. Trabalho de Conclusão de Curso, [ecomp.poli.br](http://ecomp.poli.br).

UNESP. Inovação organizacional como agente de transformação e sucesso: Estudo de caso na área de restaurantes. Disponível em: <https://www.sorocaba.unesp.br/Home/Graduacao/EngenhariadeControleeAutomacao/galdenoro1906/inovacao-organizacional-como-agente-de-transformacao-e-sucesso.pdf> (Acesso em: 28 de ago. 2025).