

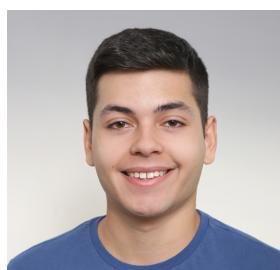
Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Desenvolvimento de Sistemas de *Software*

Ano Letivo de 2025/2026

Grupo 08

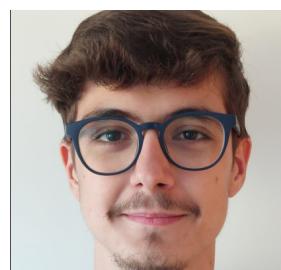
URL: <https://github.com/LEI-DSS2526/GrupoTP-08>



João Teixeira

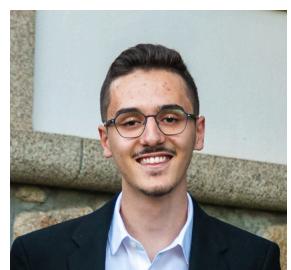
A106836

January 12, 2026



Nelson Mendes

A106884



Simão Mendes

A106928

DSS

Índice

1. Descrição dos resultados obtidos	1
2. Análise de Requisitos	2
2.1. Modelo de Domínio	2
2.2. Diagramas de Casos de Uso	5
2.2.1. UC1: Efetuar Pedido	5
2.2.2. UC2: Gerir Preparação de Pedido	5
2.2.3. UC3: Consultar Indicadores de Desempenho	5
2.2.4. UC4: Enviar Mensagem aos Colaboradores	6
2.2.5. UC5: Entregar Pedido	6
3. Modelação Conceitual Proposta	7
3.1. Diagramas de Classe	7
3.2. Diagramas de Sequência	11
4. Solução Implementada	17
4.1. Diagrama de Componentes	17
4.2. Diagramas de Classe	18
4.3. Diagramas de Sequência	22
4.4. Diagrama de <i>Packages</i>	26
5. Manual de Utilização	27
5.1. Requisitos de Sistema e Instalação	27
5.2. Configuração e Povoamento da Base de Dados	27
5.3. Guia de Navegação (Mapa de Menus)	27
5.4. Resolução de Problemas Comuns (<i>Troubleshooting</i>)	28
6. Conclusão	29
7. Bibliografia	30
8. Anexos	31
8.1. <i>Script</i> de Criação da Base de Dados	31
8.2. <i>Script</i> de População da Base de Dados	34
8.3. <i>Script</i> de Visualização da Base de Dados	40
8.4. Enunciado do Projeto	45

Lista de Figuras

Figura 1	Modelo de Domínio	3
Figura 2	Diagrama de Casos de Uso	6
Figura 3	Diagrama de Classe Conceptual geral do sistema	8
Figura 4	Diagrama de Classe Conceptual do Subsistema MenuPedido	9
Figura 5	Diagrama de Classe Conceptual do Subsistema Restaurantes	10
Figura 6	Diagrama de Sequência Conceptual - Criar Pedido	12
Figura 7	Diagrama de Sequência Conceptual - Confeccionar Pedido	13
Figura 8	Diagrama de Sequência Conceptual - Entregar Pedido	14
Figura 9	Diagrama de Sequência Conceptual - Consultar Indicadores de Desempenho	15
Figura 10	Diagrama de Sequência Conceptual - Enviar Mensagem Colaboradores	16
Figura 11	Diagrama de Componentes da Arquitetura do Sistema	17
Figura 12	Diagrama de Classe Implementado Geral do Sistema	19
Figura 13	Diagrama de Classe Implementado do Subsistema MenuPedidos	20
Figura 14	Diagrama de Classe Implementado do Subsistema Restaurantes	21
Figura 15	Diagrama de Sequência Implementado: Criação de um Pedido	23
Figura 16	Diagrama de Sequência Implementado: Ajuste de Ingredientes	23
Figura 17	Diagrama de Sequência Implementado: Consulta da Fila de Cozinha	24
Figura 18	Diagrama de Sequência Implementado: Conclusão da Preparação de um Pedido	24
Figura 19	Diagrama de Sequência Implementado: Consulta dos Pedidos Prontos	24
Figura 20	Diagrama de Sequência Implementado: Registo da Entrega de um Pedido	25
Figura 21	Diagrama de Sequência Implementado: Consulta dos Indicadores de Desempenho	25
Figura 22	Diagrama de Sequência Implementado: Envio de Mensagem aos Colaboradores	25
Figura 23	Organização Hierárquica de Pacotes Java e Dependências	26
Figura 24	Interface de Boas-vindas, Ponto de Entrada do Sistema e Início da Criação de um Pedido	28

1. Descrição dos resultados obtidos

O desenvolvimento do **Sistema de Gestão para a Cadeia de Restaurantes** resultou numa solução tecnológica robusta, capaz de automatizar o ciclo completo de operação, desde a interação inicial do cliente até à monitorização estratégica pela gestão. A implementação seguiu rigorosamente os princípios de **Engenharia de Software** lecionados, garantindo que os **requisitos funcionais e não funcionais** fossem satisfeitos de forma eficiente.

Os principais objetivos alcançados nesta solução incluem:

- **Automação do Ciclo de Pedidos:** Implementação de um fluxo contínuo onde os **Pedidos** são criados por clientes, processados em tempo real na cozinha e geridos para entrega, com atualizações de estado persistentes na **Base de Dados**. Este fluxo garante a integridade da informação e reduz a probabilidade de erros humanos no processamento de pedidos complexos;
- **Arquitetura em Camadas (Layered Architecture):** Separação clara entre a lógica de apresentação (UI), lógica de negócio (LN) e acesso a dados (DL). Esta estrutura, visível na organização dos pacotes `dss.cadeiaRestaurantesUI`, `dss.cadeiaRestaurantesLN` e `dss.cadeiaRestaurantesDL`, permitiu uma elevada **separação de responsabilidades**, o que facilitou a realização de testes unitários e a evolução independente de cada componente sem comprometer a estabilidade do sistema global;
- **Gestão Multi-Perfil e Multi-Posto:** Sistema de autenticação funcional que diferencia as capacidades de Cozinheiro, Atendente e Gestor. A interface adapta-se dinamicamente, apresentando apenas as operações relevantes ao cargo do funcionário autenticado, o que optimiza a **experiência de utilizador** e garante a segurança dos dados sensíveis;
- **Análise de Desempenho e Inteligência de Negócio:** Capacidade de extração de métricas operacionais, como volumes de **faturação** por restaurante e **tempos médios de preparação**. Esta funcionalidade oferece aos gestores uma ferramenta de suporte à decisão baseada em dados reais, permitindo identificar “*bottlenecks*” operacionais e optimizar a distribuição de recursos;
- **Escalabilidade e Persistência:** O sistema foi desenhado para suportar o crescimento da cadeia, permitindo a adição de novos Restaurantes e Produtos com impacto mínimo no código existente, graças à utilização de padrões de desenho como o **DAO** (*Data Access Object*) e o **Facade** (implementado em `CadeiaRestaurantesLNFacade`).

2. Análise de Requisitos

Nesta fase é estabelecido o alicerce essencial para a construção de um sistema coerente e alinhado com as necessidades reais dos *stakeholders*. Este estágio, sucede o entendimento inicial do problema e dedica-se a uma **investigação meticulosa e estruturação do conhecimento** adquirido. O objetivo central é **capturar, especificar e organizar** as funcionalidades, regras e conceitos críticos do negócio, de modo a traduzir a visão do cliente e dos utilizadores finais em uma especificação clara e não ambígua. O trabalho realizado aqui materializa-se em dois artefactos fundamentais que guiarão todo o *design* subsequente:

- a criação do **Modelo de Domínio**, que descreve visualmente as entidades, os seus respetivos atributos e relações no universo do problema;
- a definição detalhada dos **Casos de Uso**, que encapsulam as interações entre atores e o sistema. Estes, por sua vez, serão representados através de **Diagramas de Casos de Uso** e tem por objetivo servir como um contrato comportamental que direciona o desenvolvimento das funcionalidades e a modelagem dinâmica do sistema.

Assim, uma **Análise de Requisitos** robusta é a premissa para um Modelo de Domínio preciso e para Casos de Uso bem definidos, os quais, em conjunto, formam a bússola para as fases de **design, implementação e validação**.

2.1. Modelo de Domínio

O Modelo de Domínio desenvolvido teve como principal objetivo representar, de forma fiel e estruturada, a realidade de funcionamento de uma cadeia de restaurantes de *fast-food*, conforme descrita no enunciado do trabalho. A seguinte abordagem **não se limitou a identificar entidades óbvias**, como o Pedido ou o Cliente, mas procurou **capturar os conceitos necessários** para suportar tanto o funcionamento operacional do sistema como a recolha de informação relevante para a gestão da cadeia:

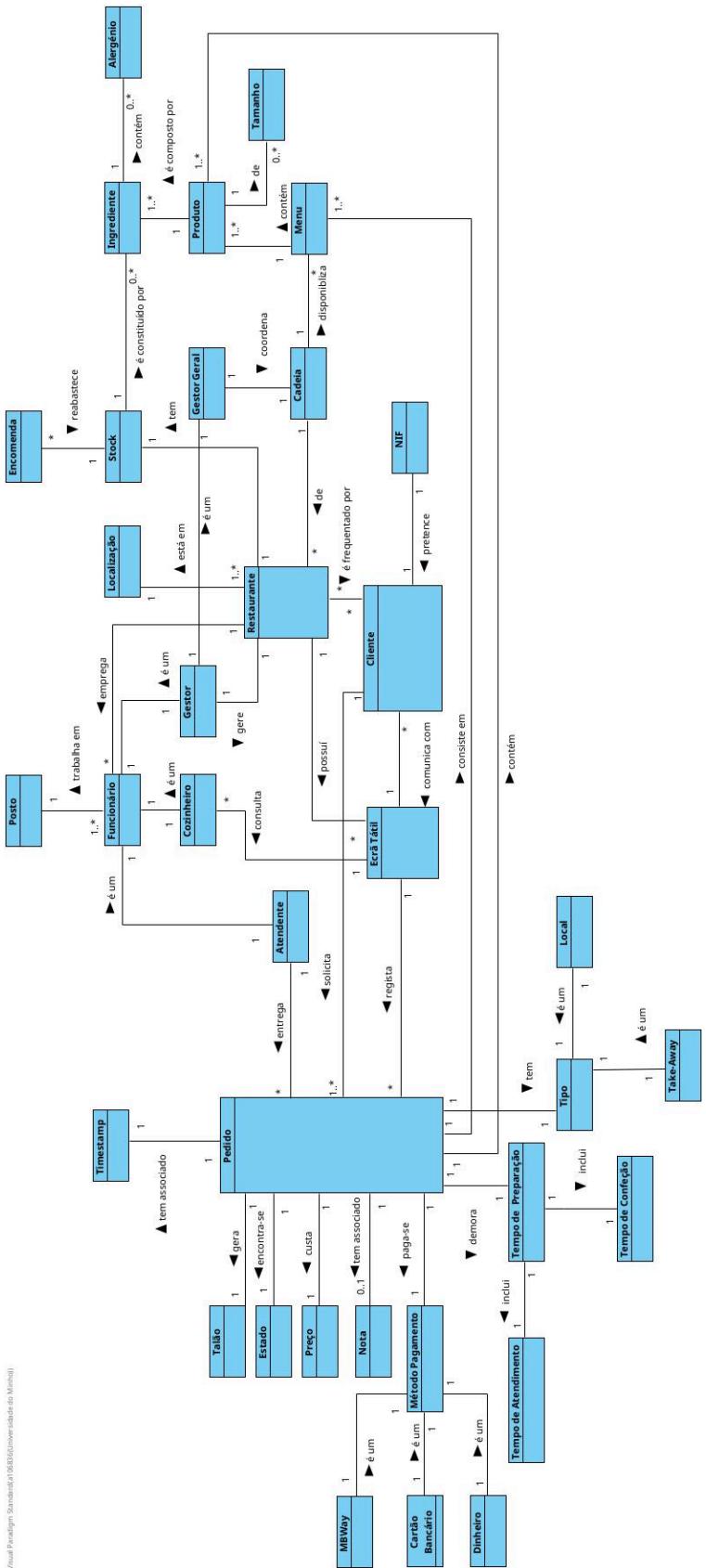


Figura 1: Modelo de Domínio.

O **Pedido** foi eleito como elemento central, pois conecta todos os processos (do Cliente ao Cozinheiro e Gestor), agregando informações como **Estado**, **Preço**, **Tempo** e **Método de Pagamento**, essenciais para indicadores de desempenho.

Para suportar a personalização, foram modeladas entidades distintas:

- **Produto** (item individual) e **Menu** (conjunto de produtos);
- **Ingrediente** (componente personalizável) e **Alergénio** (associado a ingredientes para validação automática);
- **Tamanho**, para representar opções como volumes de bebidas.

A gestão operacional foi contemplada com:

- **Stock**, **Encomenda** e **Ingrediente**, para fechar o ciclo entre consumo e reposição;
- Diferentes tipos de **Tempo** (**atendimento**, **preparação**, **confeção**) para análise de desempenho;
- **Funcionário com especializações** (Cozinheiro, Atendente, Gestor) para refletir papéis e responsabilidades.

A distinção entre **Cadeia** e **Restaurante** surge da necessidade de suportar uma visão global do negócio. O sistema destina-se a uma cadeia de restaurantes e não a uma unidade isolada, pelo que é fundamental conseguir **agregar informação ao nível da cadeia**, comparar restaurantes e analisar o desempenho em diferentes localizações. Esta **estrutura hierárquica** permite responder diretamente aos requisitos de informação para a gestão descritos no enunciado.

Por fim, o **Ecrã Tátil** foi incluído como entidade do domínio, pois desempenha um papel central no funcionamento do sistema. Apesar de ser um elemento tecnológico, é através dele que o **cliente interage com o sistema, efetua pedidos, transmite notas e escolhas, e inicia todo o processo de produção**. No contexto deste sistema, o ecrã tátil não é apenas uma interface, mas um elemento funcional que influencia diretamente o fluxo do pedido, justificando a sua presença no modelo.

Em síntese, o **Modelo de Domínio** desenvolvido não resulta de uma tentativa de adicionar complexidade desnecessária, mas sim de tornar explícitos **conceitos que estão implícitos no enunciado** e que são essenciais para suportar os objetivos do sistema. As **entidades e relações adicionais** permitem representar corretamente a personalização dos pedidos, a gestão de alergénios, a organização do trabalho na cozinha, o controlo de stocks e a produção de indicadores relevantes para a gestão da cadeia de restaurantes.

2.2. Diagramas de Casos de Uso

Os **Diagramas de Casos de Uso** foram elaborados com o objetivo de identificar e estruturar as funcionalidades do sistema do ponto de vista dos diferentes intervenientes, de acordo com os requisitos definidos no enunciado. Estes diagramas permitem representar as **interações entre os atores e o sistema**, evidenciando as principais ações que cada tipo de utilizador pode realizar e os limites de responsabilidade do sistema. A sua construção teve como base os cenários descritos para o funcionamento dos restaurantes, desde a **realização do pedido** pelo cliente até à **sua confecção, entrega e apoio à gestão**, de modo a garantir uma visão funcional clara e coerente que serve de suporte às fases seguintes de modelação e implementação.

2.2.1. UC1: Efetuar Pedido

Este caso de uso descreve o processo principal pelo qual um **cliente interage com o sistema para fazer uma encomenda**. O cliente utiliza um ecrã tátil no restaurante para **selecionar produtos**, fornecendo a possibilidade de personalização e adição de notas especiais, e **finalizar o pagamento** através de múltiplos métodos. Após o pagamento bem-sucedido, o cliente **indica se irá consumir no local ou levar para fora (takeaway)**, e o sistema regista o pedido e emite um talão com um número e o tempo estimado de espera. Este UC é fundamental, pois é o ponto de entrada de todos os pedidos no sistema.

2.2.2. UC2: Gerir Preparação de Pedido

Este caso de uso cobre as atividades dos cozinheiros para preparar os pedidos recebidos. O sistema apresenta uma **fila de pedidos com tempos estimados**. O cozinheiro seleciona um pedido para preparar, o que **altera o seu estado para “em preparação”**. Após a conclusão, o cozinheiro marca-o como “pronto para entrega”, o sistema atualiza o estado, remove-o da sua fila e notifica o cliente via *display*. Os **fluxos alternativos** tratam de situações problemáticas:

- se um **ingrediente em falta** estiver disponível no armazém, o pedido pode ser reagendado, recalculando os tempos de espera e notificando o cliente;
- se o **ingrediente estiver indisponível** em stock, o pedido é cancelado e o cliente é notificado.

2.2.3. UC3: Consultar Indicadores de Desempenho

Este caso de uso permite que **um gestor** (de um restaurante específico ou geral da cadeia) **analise métricas de negócio, como faturação e desempenho operacional**. O sistema identifica automaticamente o restaurante do gestor e **apresenta os indicadores correspondentes**. Se o utilizador **for um Gestor Geral**, o sistema primeiro permite-lhe selecionar um ou vários restaurantes da cadeia para agregar ou comparar dados. Com base na análise, o **gestor pode decidir não tomar ação ou proceder com uma comunicação interna** com a equipa, invocando o UC4.

2.2.4. UC4: Enviar Mensagem aos Colaboradores

Este caso de uso suporta a comunicação interna dentro do restaurante ou da cadeia. Um gestor (local ou geral) pode redigir uma mensagem e enviá-la para os *displays* dos postos de trabalho selecionados, como por exemplo: cozinha e caixa. A mensagem é exibida em tempo real e fica registada no histórico. Um Gestor Geral tem a capacidade adicional de enviar a mesma mensagem para os postos de trabalho de múltiplos restaurantes.

2.2.5. UC5: Entregar Pedido

Este caso de uso finaliza o ciclo do cliente e foca-se na entrega física do pedido. Quando um pedido está marcado como “pronto para entrega”, aparece na interface do atendente e num *display* público. O atendente entrega o pedido ao cliente, confirma a entrega no sistema, que atualiza o estado do pedido para “Entregue”, remove-o da lista de pendentes e regista o evento. Este UC garante o fecho formal do processo e a precisão dos dados históricos.

Em síntese, estes casos de uso mapeiam de forma coerente o fluxo principal de valor do sistema: desde a **criação do pedido** pelo cliente (UC1), passando pela **sua preparação** na cozinha (UC2) e **entrega final** (UC5), até às **funções de gestão e monitorização** que suportam a operação (UC3 e UC4). Segue-se assim o Diagrama de Casos de Uso:

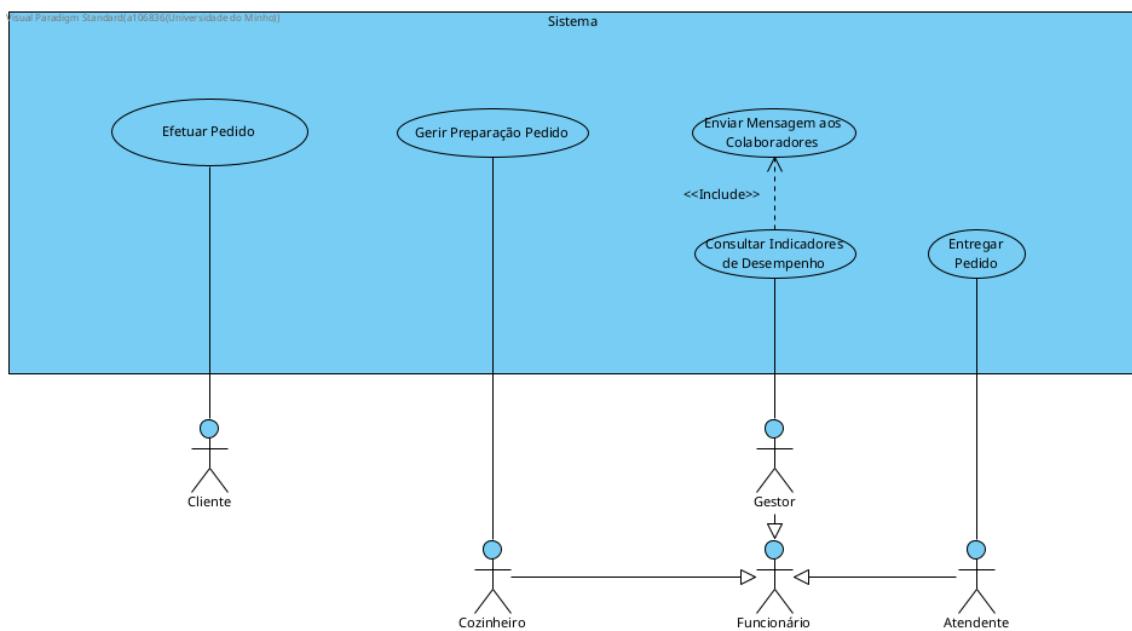


Figura 2: Diagrama de Casos de Uso.

3. Modelação Conceitual Proposta

A **modelação conceptual** constitui uma etapa fundamental no processo de desenvolvimento, este permite transpor os requisitos identificados na fase de análise para uma **representação abstrata e independente da tecnologia**. Nesta fase, o foco recai sobre a definição da estrutura lógica e do comportamento dinâmico que o sistema deve suportar para satisfazer as necessidades de automatização da cadeia de restaurantes.

3.1. Diagramas de Classe

Os **diagramas de classe conceptuais** detalham a estrutura estática do sistema, mapeando entidades, atributos e relações de associação, agregação e herança. Este modelo assegura a integridade dos dados e a correta segmentação entre produtos individuais e menus compostos.

Explicação dos Subsistemas: O sistema foi logicamente dividido em dois subsistemas principais para garantir uma separação clara de responsabilidades e facilitar a futura implementação multi-camada:

- **Subsistema de Restaurantes:** Gere a infraestrutura e os recursos humanos no nível de cada restaurante. Define a **hierarquia de Funcionário** (Gestor, Cozinheiro e Atendente) e a sua alocação aos Postos de trabalho especializados (e.g., grelha, fritos). Este subsistema é também responsável pela monitorização de indicadores de desempenho e gestão de fluxos de trabalho;
- **Subsistema de Menu e Pedidos:** Este subsistema **gere o fluxo transacional** e o catálogo de venda disponível nos ecrãs táteis. Define a relação entre Pedidos, Linhas de Pedido e os Artigos de Venda, que podem apresentar-se como Produtos individuais ou Menus já compostos. O **modelo prevê a lógica de personalização**, permitindo adicionar ou retirar Ingredientes e validar opções de confeção de acordo com as preferências ou alergénios indicados pelo cliente. Garante ainda que as **notas especiais** de cada pedido **sejam visualizadas nos displays dos postos de trabalho** relevantes.

Segue-se a representação dos diagrama de classes propostos:

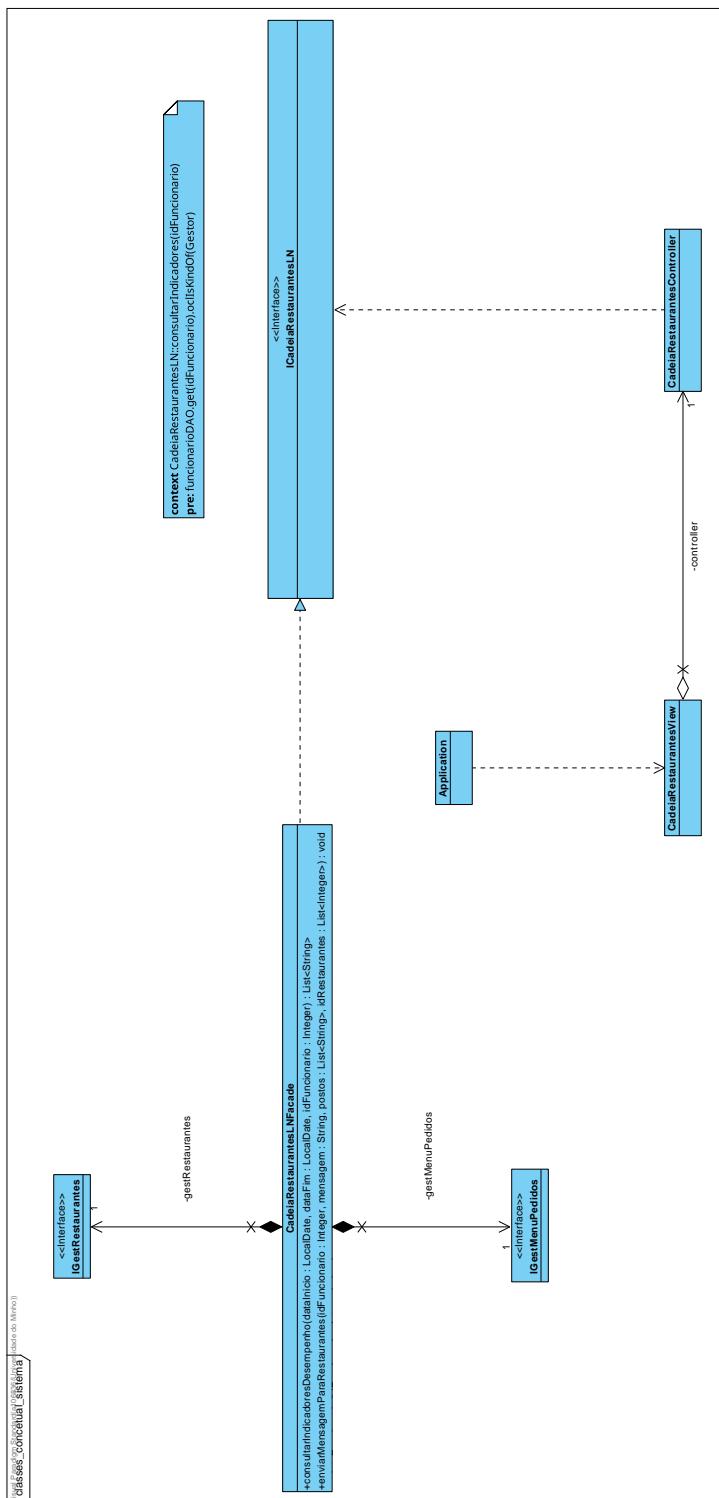
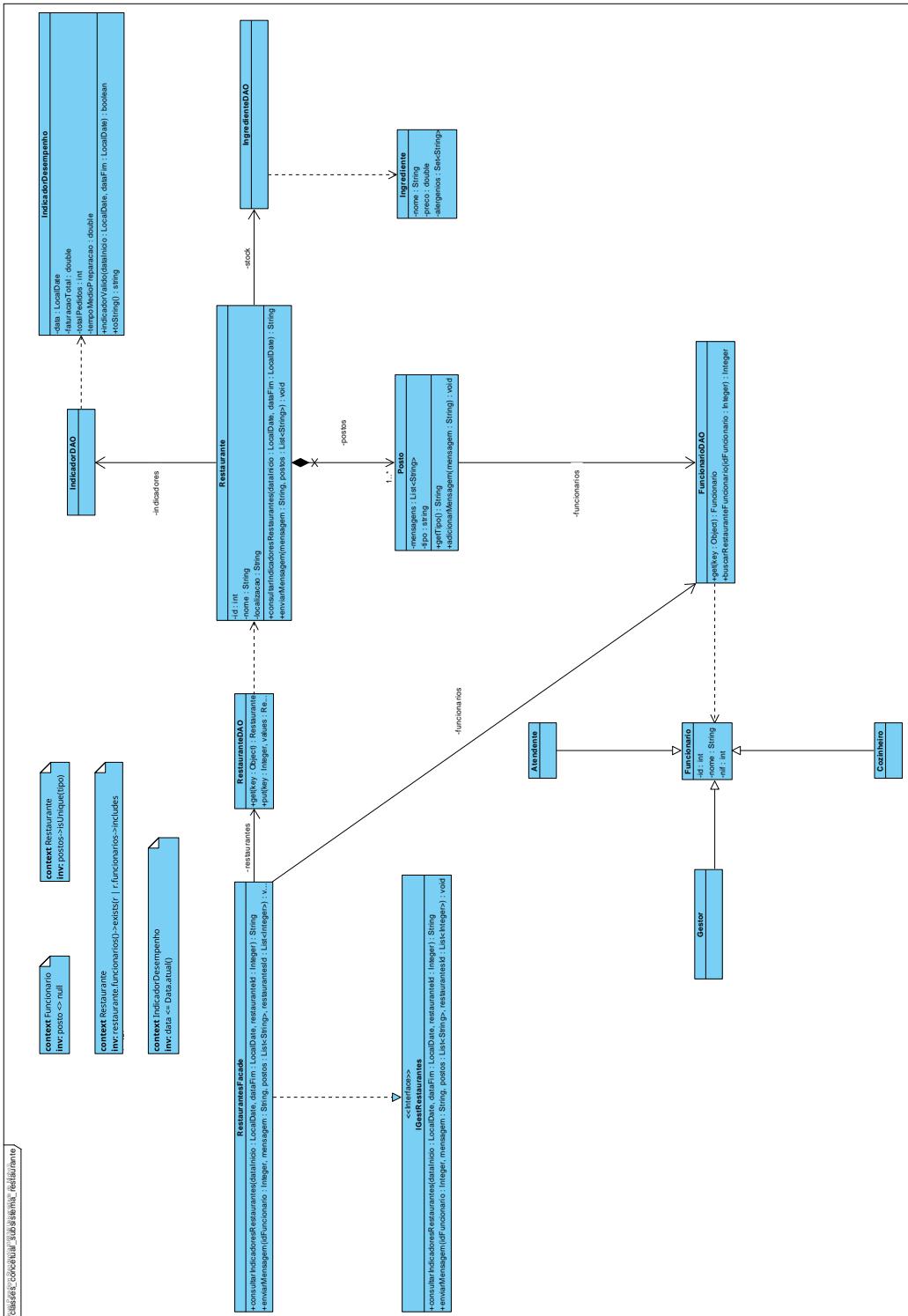


Figura 3: Diagrama de Classe Conceptual geral do sistema.





A articulação entre os subsistemas apresentados permite uma gestão integrada do ciclo de vida de cada pedido, desde o auto-atendimento até à produção. É importante notar que a modelação conceptual aqui exposta já contempla a integração de padrões de persistência, nomeadamente os *Data Access Objects* (DAO). Esta decisão deriva do facto de termos iniciado a segunda fase do projeto num período em que os conteúdos programáticos relativos a esta arquitetura já tinham sido lecionados, permitindo uma transposição mais madura e próxima da implementação final.

Neste modelo conceptual, encontram-se transpostos apenas os métodos estritamente necessários à execução dos Diagramas de Sequência apresentados na secção seguinte.

A integração destes subsistemas permite um fluxo de informação contínuo desde o terminal de atendimento até à entrega final. A **estrutura polimórfica** adotada para os artigos de venda (ArtigoVenda) facilita a gestão de inventário e a **escalabilidade** do catálogo, enquanto a modelação dos postos de trabalho permite a reordenação dinâmica de tarefas perante imprevistos operacionais.

3.2. Diagramas de Sequência

Os **diagramas de sequência** descrevem o comportamento dinâmico do sistema e ilustram a troca de mensagens entre objetos para concretizar os cenários de utilização. As operações detalhadas nestes diagramas foram **inferidas diretamente dos casos de uso** identificados na fase de análise e utilizam estritamente a estrutura definida no diagrama de classes conceptual. O foco destas representações recai na coordenação entre a interface de atendimento, o processamento de pedidos e os mecanismos de persistência (**DAO**), o que **garante a integridade da informação** desde o terminal até à entrega final. Esta abordagem valida a suficiência dos métodos propostos no modelo conceitual para realizar as operações de registo, confeção e gestão da cadeia de restaurantes.

Apresentam-se de seguida a descrição dos diagramas correspondentes aos fluxos principais do sistema:

- **Criar Pedido:** Detalha o fluxo de criação de um pedido pelo cliente, descrevendo a seleção e personalização de produtos, a validação de ingredientes e a finalização da transação através dos diversos métodos de pagamento suportados;
- **Confeccionar Pedido:** Ilustra o processo de produção, focando-se na gestão das filas de trabalho nos postos especializados e na interação dos cozinheiros com o sistema para a atualização do estado dos itens;
- **Entregar Pedido:** Foca-se na fase final do ciclo de vida do pedido, detalhando o registo da entrega ao cliente e a consequente atualização do estado para “Entregue” no sistema;

- **Consultar Indicadores de Desempenho:** Descreve o fluxo de consulta de métricas de gestão, onde o sistema agrupa dados de produtividade e tempos de espera para apoio à tomada de decisão;
- **Enviar Mensagem Colaboradores:** Ilustra o mecanismo de comunicação interna, o que permite o envio de instruções ou notas especiais entre a gestão e os diversos postos de trabalho.

Seguem-se os diagramas de sequência conceptualizados realizados:

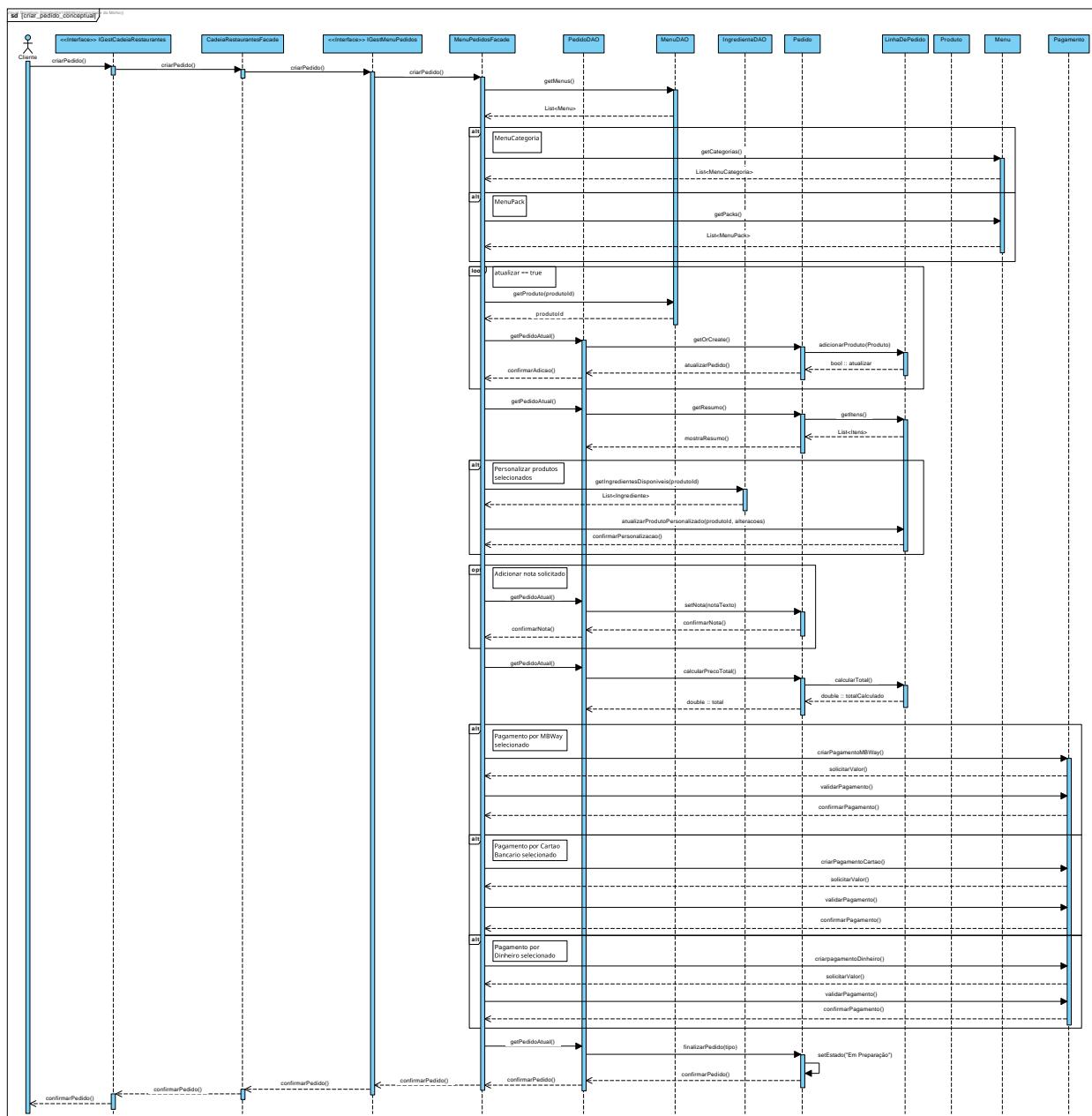


Figura 6: Diagrama de Sequência Conceptual - Criar Pedido.

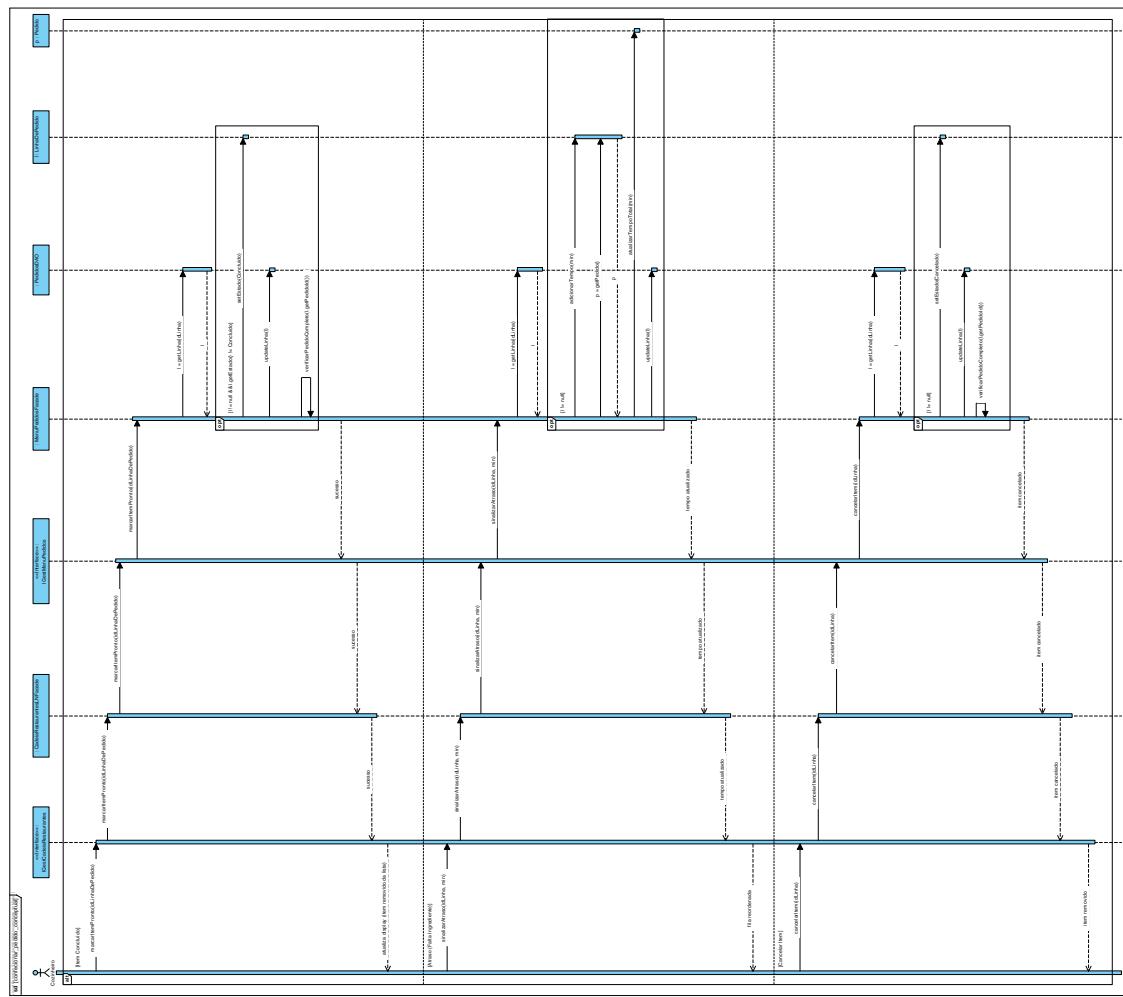


Figura 7: Diagrama de Sequência Conceptual - Confeccionar Pedido.

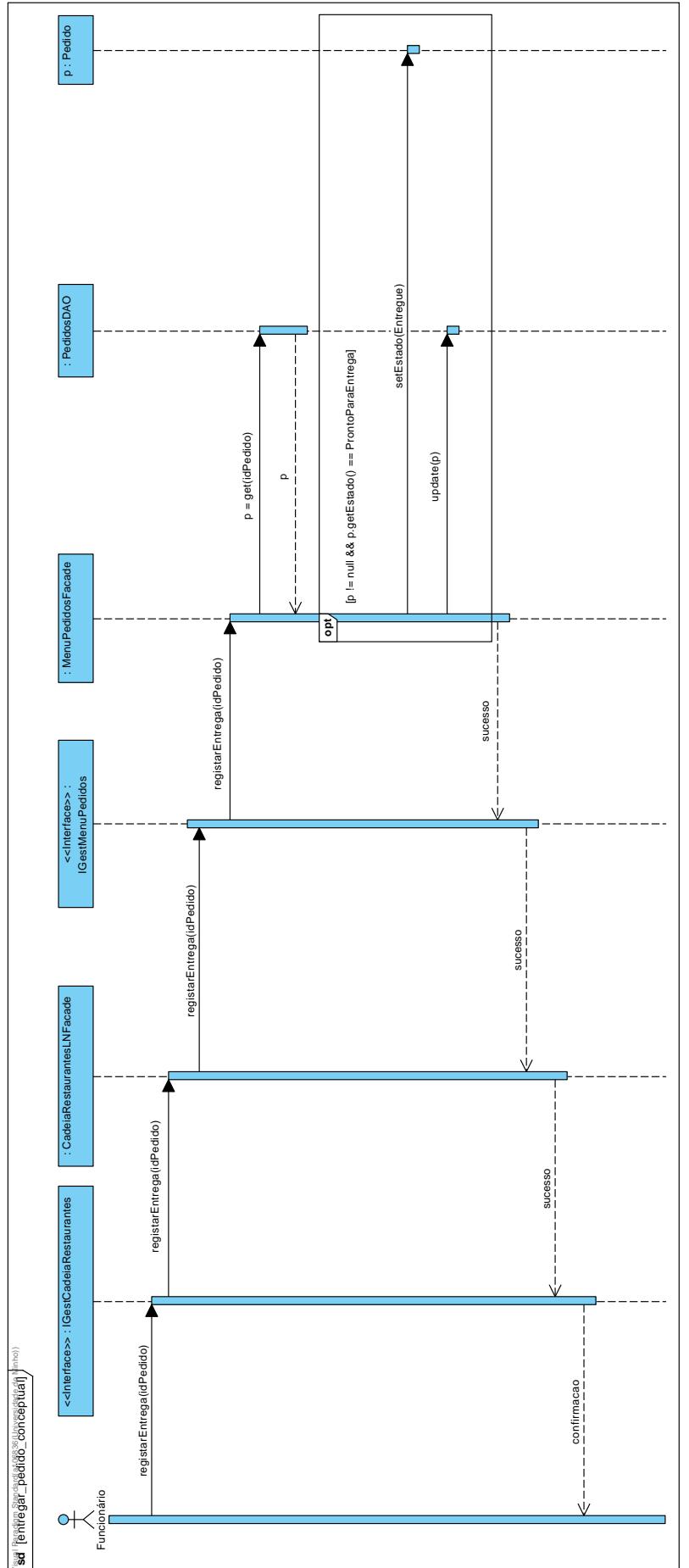


Figura 8: Diagrama de Sequência Conceptual - Entregar Pedido.

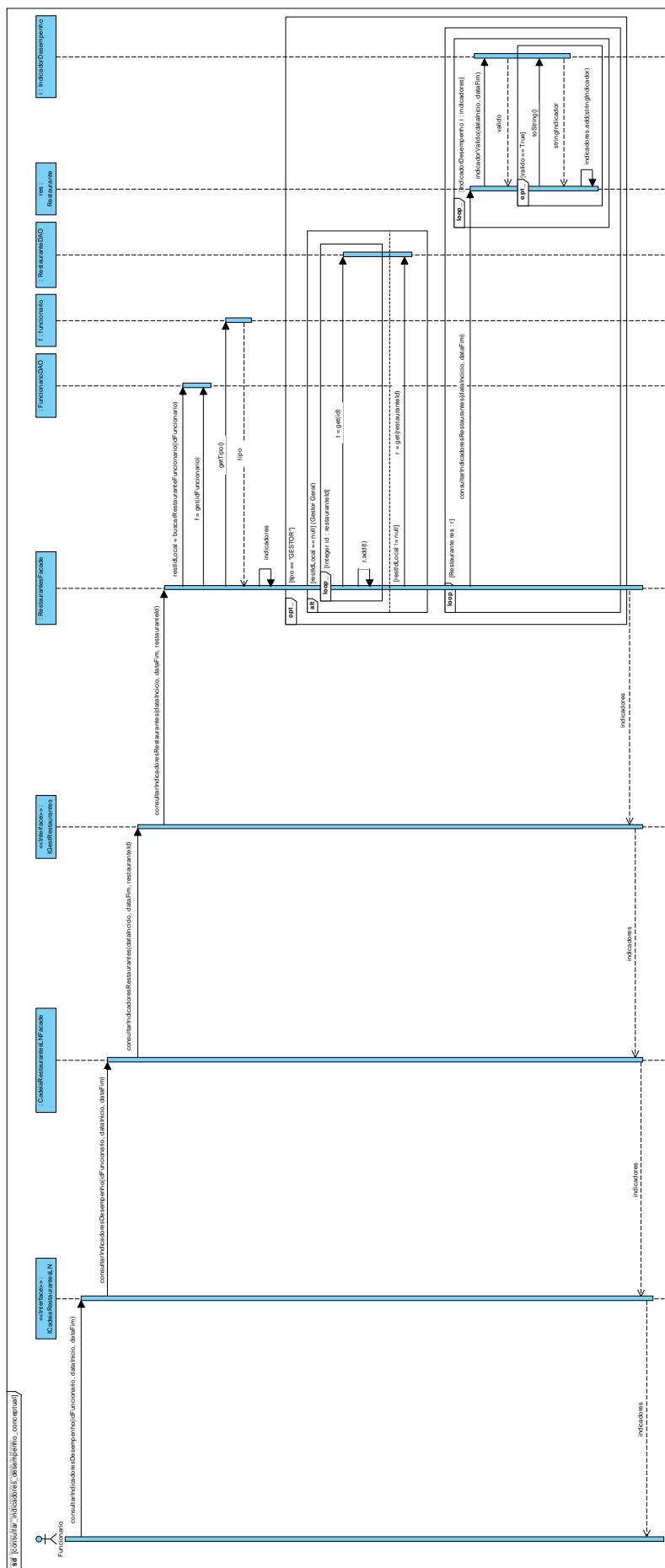


Figura 9: Diagrama de Sequência Conceptual - Consultar Indicadores de Desempenho.

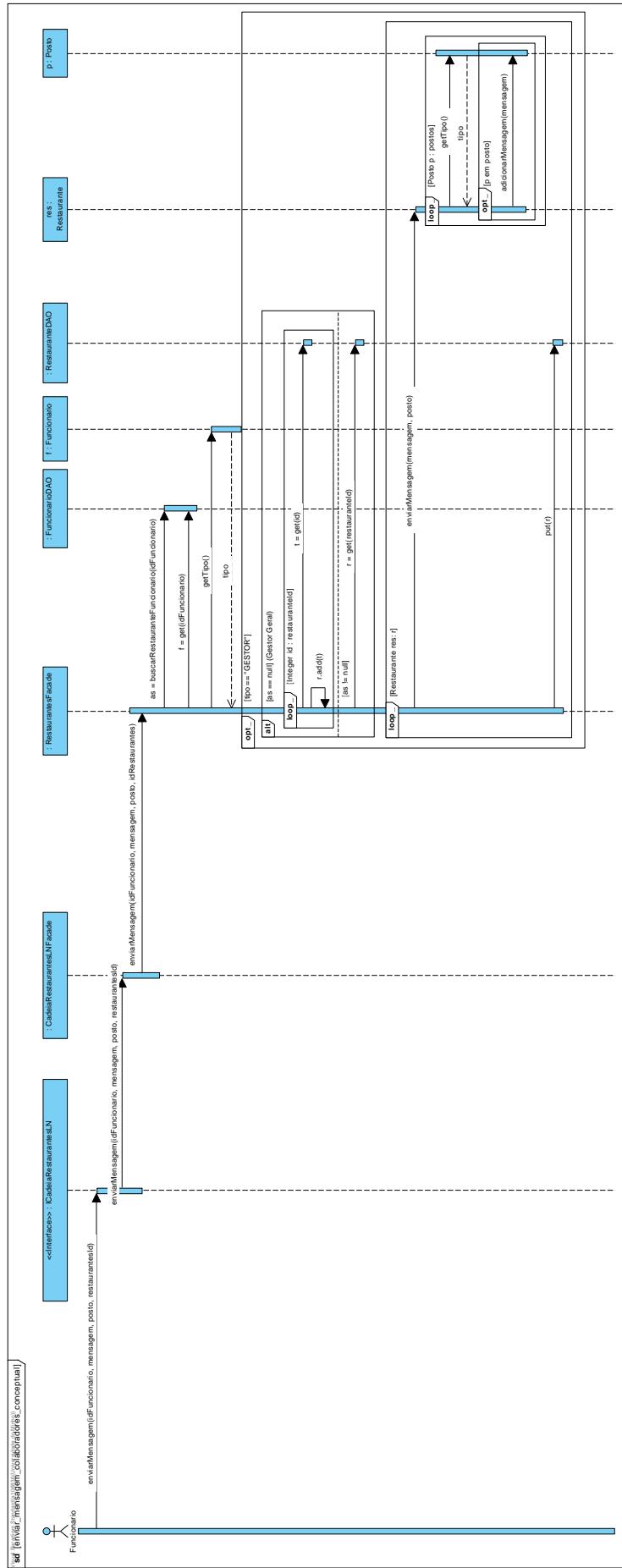


Figura 10: Diagrama de Sequência Conceptual - Enviar Mensagem Colaboradores.

4. Solução Implementada

4.1. Diagrama de Componentes

O **diagrama de componentes** detalha a decomposição modular do sistema, evidenciando as dependências lógicas e as interfaces de comunicação entre os diferentes blocos funcionais. A arquitetura implementada tira partido da **modularização** para isolar as responsabilidades de interface, lógica e persistência:

- **Componente UI (CadeiaRestaurantesUI):** Representa a **camada de apresentação** que interage com o utilizador. Este componente depende exclusivamente da interface ICadeiaRestaurantesLN, garantindo que a interface permanece agnóstica em relação às implementações internas da **lógica de negócio**;
- **Componente LN (CadeiaRestaurantesLN):** Atua como o núcleo do sistema, implementando a **fachada principal** (através da classe CadeiaRestaurantesLNFacade) e orquestrando dois subsistemas especializados:
 - **SubSistemaMenuPedidos:** Gere o catálogo de venda e o processamento transacional, expondo a interface IGestMenuPedidos;
 - **SubSistemaRestaurantes:** Responsável pela gestão operacional das unidades e funcionários, expondo a interface IGestRestaurantes;
- **Camada de Persistência (DAOs):** O acesso aos dados é realizado através de componentes especializados para cada entidade (como ProdutoDAO, PedidoDAO, MenuDAO, IngredienteDAO, RestauranteDAO, IndicadorDAO e FuncionarioDAO). Estes componentes são responsáveis pela sincronização entre as estruturas de dados em memória (mapeamentos do tipo Map<Integer, T>) e a base de dados MariaDB.

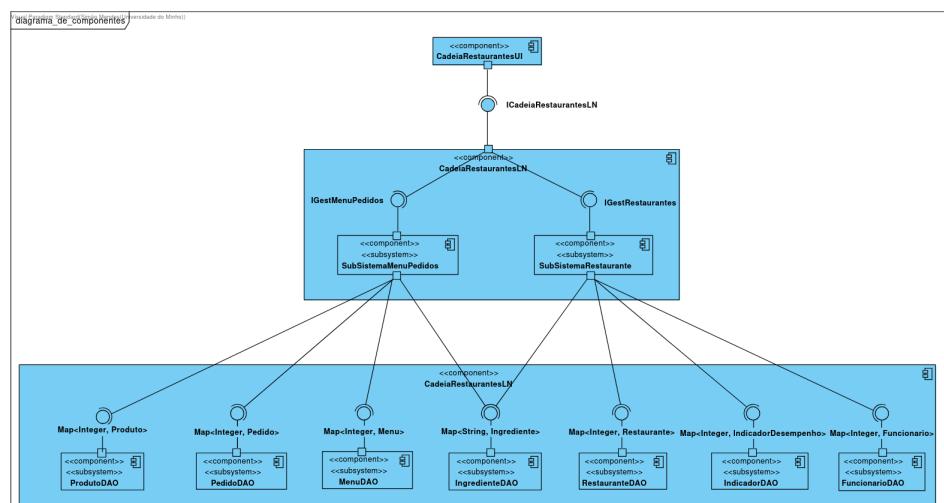


Figura 11: Diagrama de Componentes da Arquitetura do Sistema.

4.2. Diagramas de Classe

Os **diagramas de classe** da solução implementada descrevem a arquitetura final do *software* e concretizam o modelo conceitual através de padrões de desenho orientados a objetos. A implementação adota uma estrutura multi-camada organizada nos pacotes `cadeiaRestaurantesUI` (apresentação), `cadeiaRestaurantesLN` (lógica de negócios) e `cadeiaRestaurantesDL` (acesso a dados).

Estes diagramas apresentam:

- **Visão Geral da Arquitetura:** Ilustra a organização do projeto em camadas e a utilização do padrão Facade (`CadeiaRestaurantesLN`) como ponto de entrada único para a lógica de negócios;
- **Subsistema de Menu e Pedidos:** Detalha a implementação de classes como `Pedido`, `ArtigoVenda` e `Produto`, incluindo o **suporte polimórfico** para os diferentes métodos de Pagamento;
- **Subsistema de Restaurantes e Gestão:** Descreve a estrutura técnica de entidades como `Restaurante`, `Posto` e a hierarquia de `Funcionario`;
- **Camada de Persistência (*Data Layer*):** Expõe a estrutura dos *Data Access Objects* (DAO), que gerem a comunicação com a base de dados e asseguram a persistência dos objetos.

De notar que, para evitar a **poluição visual dos diagramas** e manter o **foco na estrutura lógica**, não foram incluídos métodos utilitários como `toString()`, getters ou setters que não possuam relevância direta para a compreensão das interações, apesar de os mesmos estarem presentes na implementação do código.

Apresentam-se de seguida os diagramas que detalham a estrutura técnica do sistema:

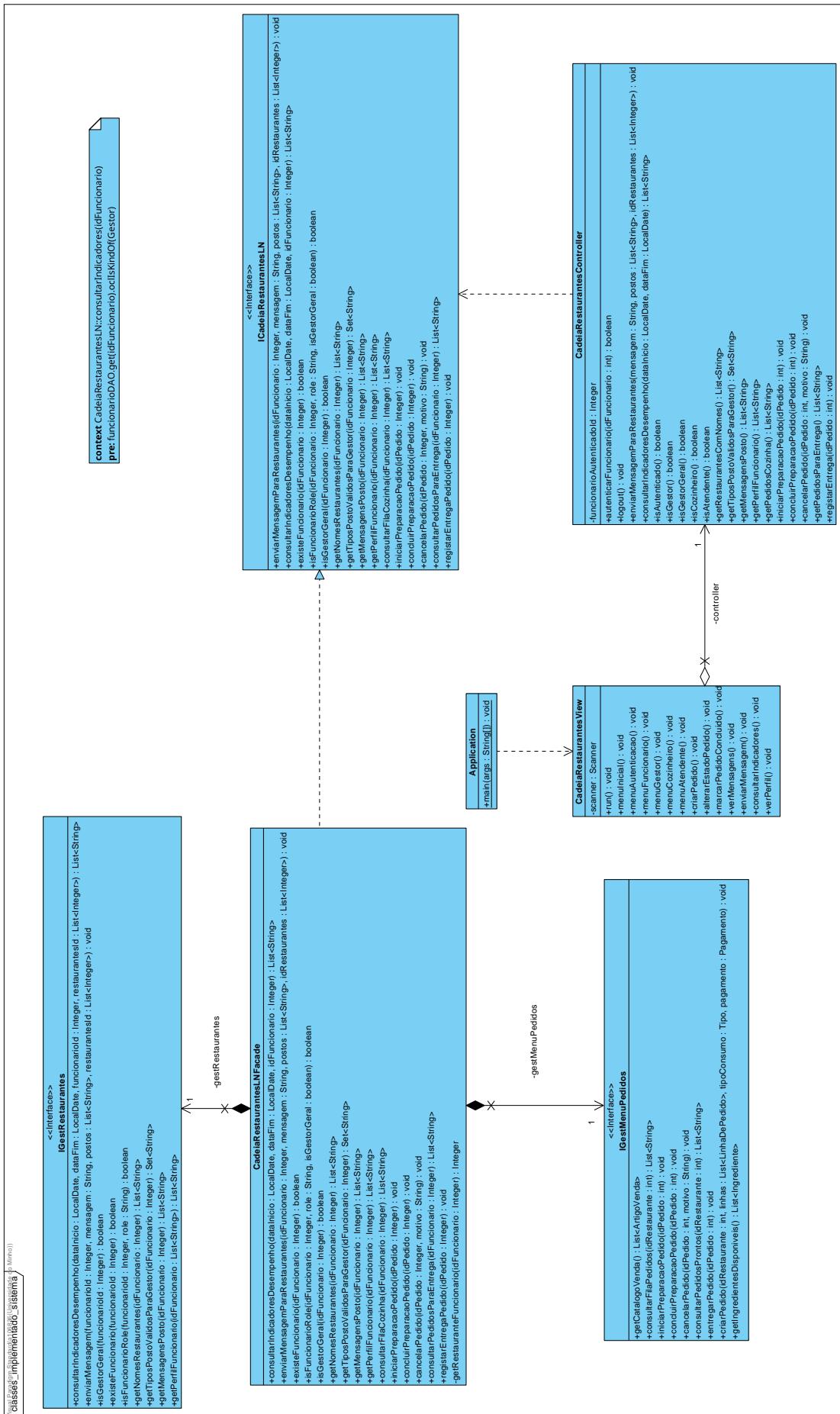


Figura 12: Diagrama de Classe Implementado Geral do Sistema.

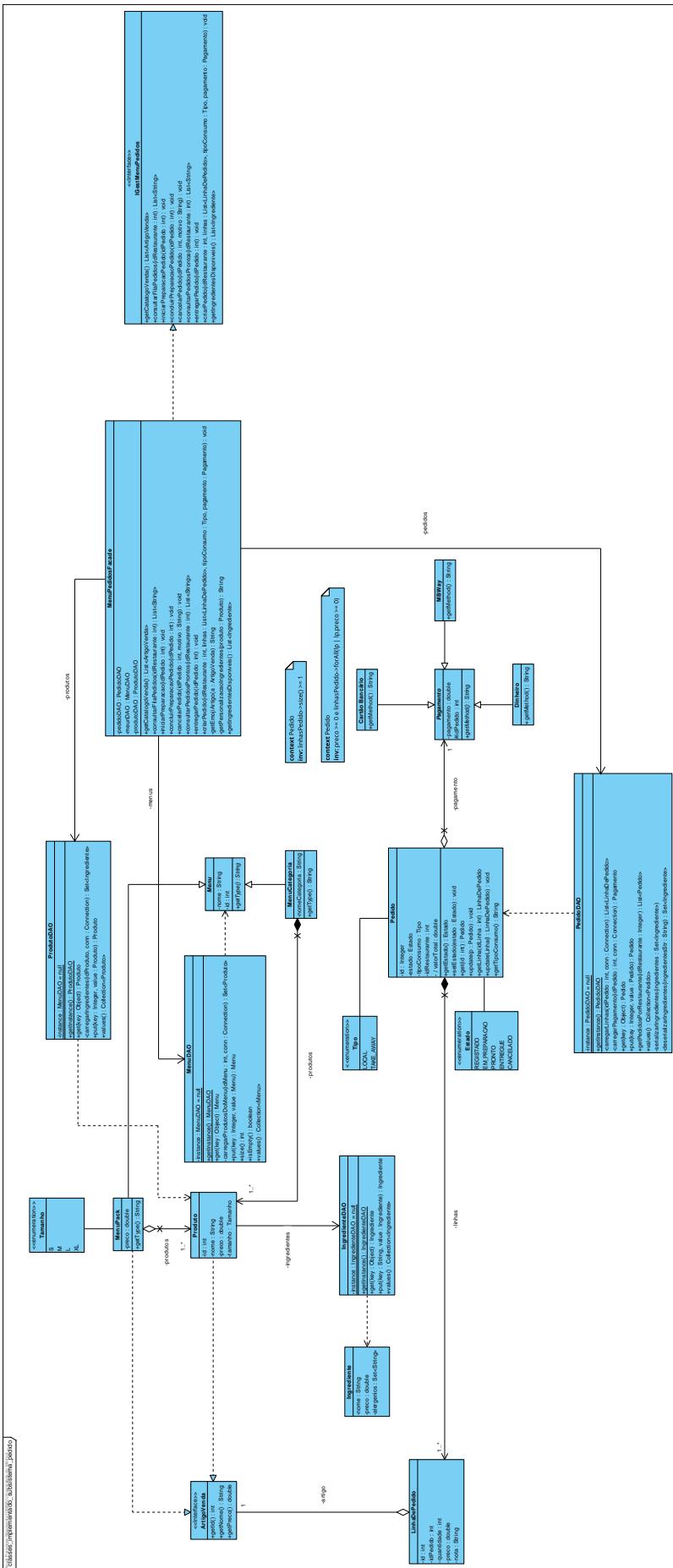


Figura 13: Diagrama de Classe Implementado do Subsistema MenuPedidos.

4.3. Diagramas de Sequência

Os **diagramas de sequência** implementados têm como objetivo ilustrar, de forma clara e estruturada, o comportamento dinâmico do sistema com base no código efetivamente implementado. Estes diagramas evidenciam a **interação entre o cliente e o sistema desenvolvido**, servindo, assim, como um complemento essencial à documentação do sistema, **facilitando a validação da implementação** face aos requisitos funcionais definidos.

Os dois diagramas referentes ao **CU1** (**Figura 15** e **Figura 16**) representam, em conjunto, o **processo completo de criação e personalização de um pedido**, integrando o ciclo iterativo de ajuste de ingredientes de um produto com o fluxo global do sistema, desde a seleção do restaurante e consulta do catálogo, **passando pela personalização, adição de notas** e ajuste de custos, até à escolha do método de pagamento e criação final do pedido, assegurando a **coerência entre a interação** do utilizador, as regras de negócio e a persistência de dados.

Os quatro diagramas referentes ao **CU2** (derivados dos 2 diagramas concetuais `entregar_pedido` e `confeccionar_pedido`) descrevem fluxos específicos de um **sistema de gestão de pedidos**:

- a **Figura 17, o acesso do cozinheiro à fila de pedidos** registados ou em preparação;
- a **Figura 18** mostra a **conclusão da preparação** pelo cozinheiro, com validação e atualização do estado;
- a **Figura 19, a visualização pelo atendente dos pedidos** prontos para entrega;
- a **Figura 20, o registo da entrega**, com validação do estado **PRONTO** antes da sua atualização para **ENTREGUE**.

Todos seguem uma arquitetura em camadas (**View → Controller → Facade → DAO**), incorporando validações de estado e tratamento de exceções.

Os dois diagramas referentes aos **CU3** e **CU4** detalham funcionalidades adicionais do **sistema de gestão de restaurantes**:

- a **Figura 21** ilustra o processo de **consulta de indicadores de desempenho** por um Gestor, no qual o sistema obtém o restaurante associado, calcula e formata os indicadores, retornando a lista consolidada;
- a **Figura 22** mostra o **fluxo de envio de mensagens**, onde o sistema recupera o restaurante do Gestor, obtém os seus postos e envia uma mensagem individual para cada posto, adicionando-as à fila de mensagens para distribuição.

Ambos os fluxos operam numa arquitetura em camadas e envolvem processos iterativos de dados. Também é relevante referir que se fosse um **Gestor Geral**, os diagramas ilustrariam o processo relativo a **todos os restaurantes da cadeia**.

Com isto seguem-se os **Diagramas de Sequência Implementados**:

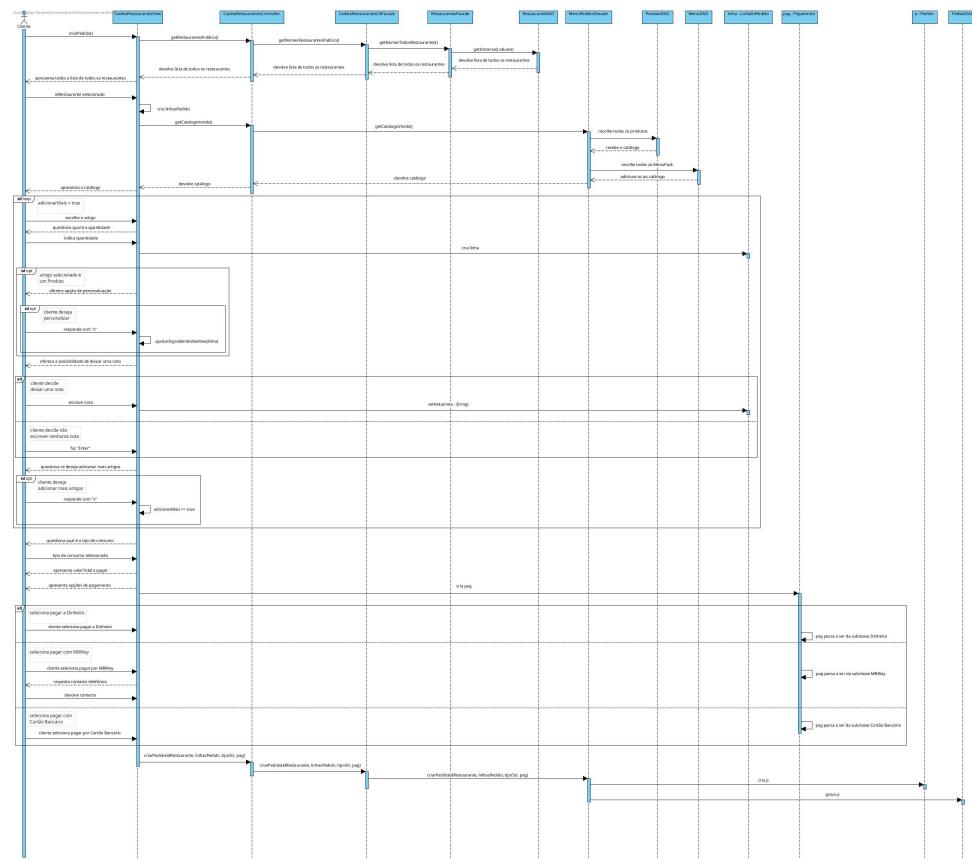


Figura 15: Diagrama de Sequência Implementado: **Criação de um Pedido**.

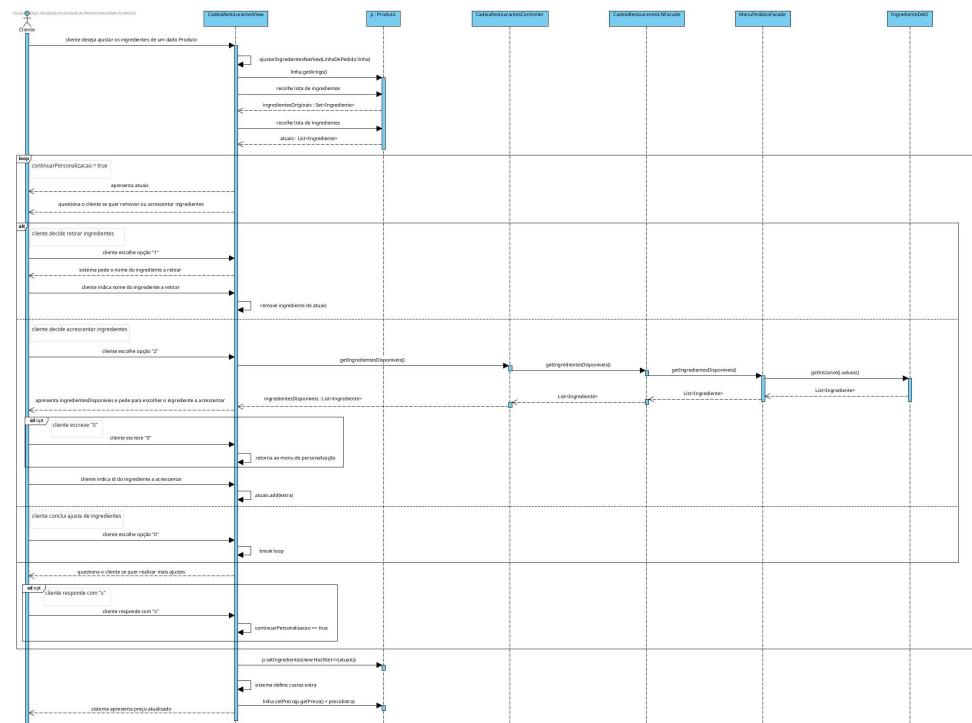


Figura 16: Diagrama de Sequência Implementado: **Ajuste de Ingredientes**.

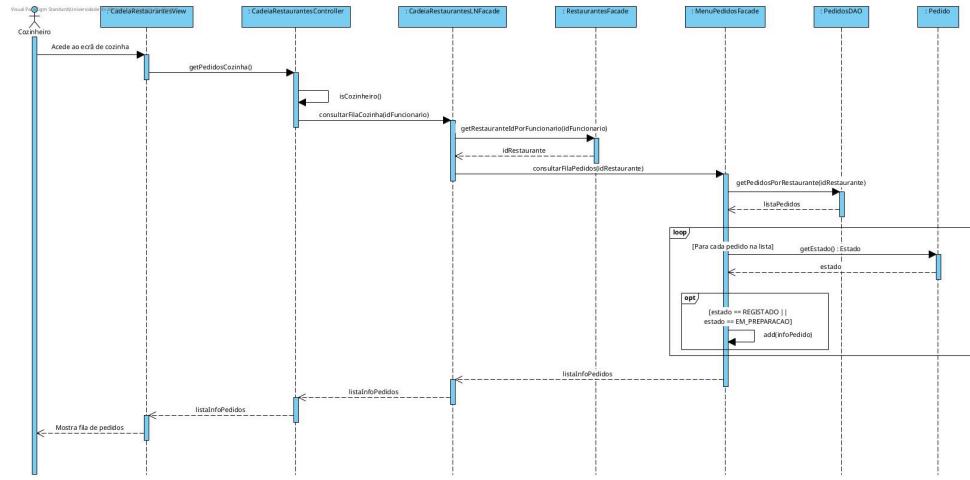


Figura 17: Diagrama de Sequência Implementado: **Consulta da Fila de Cozinha**.

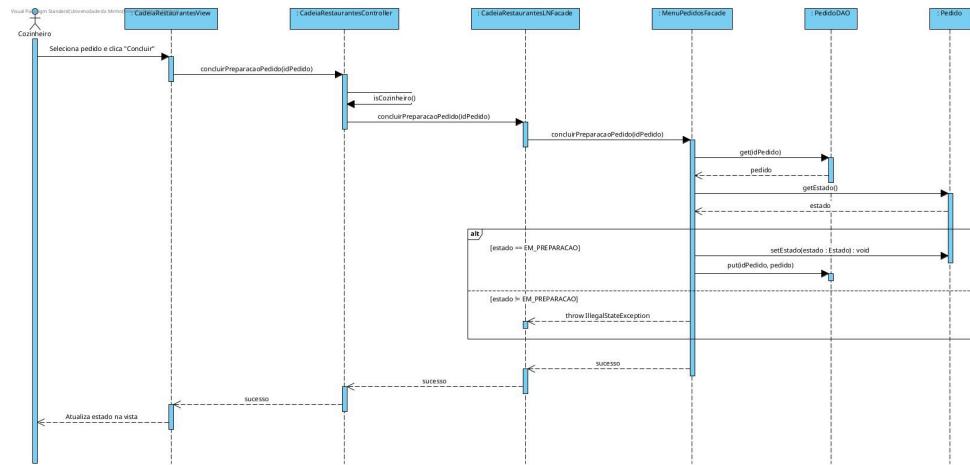


Figura 18: Diagrama de Sequência Implementado: **Conclusão da Preparação de um Pedido**.

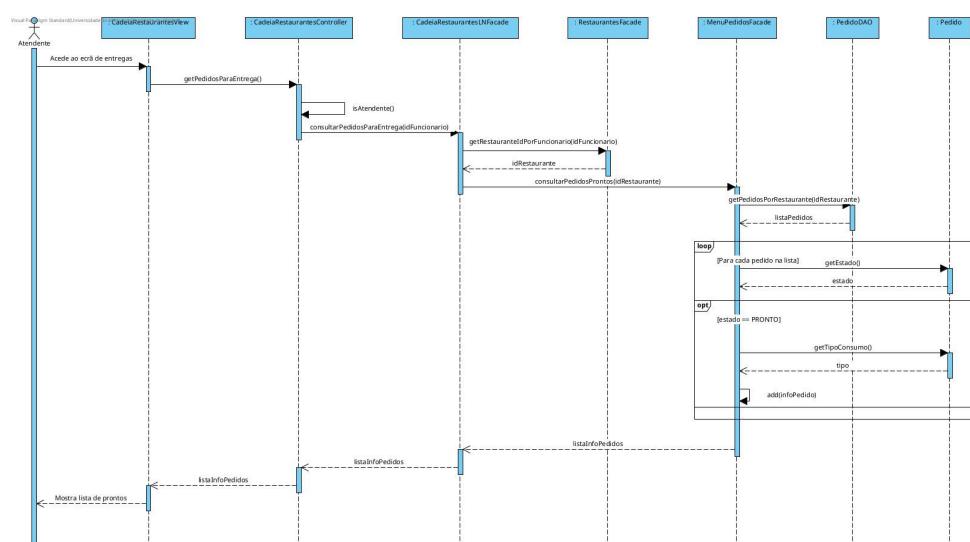


Figura 19: Diagrama de Sequência Implementado: **Consulta dos Pedidos Prontos**.

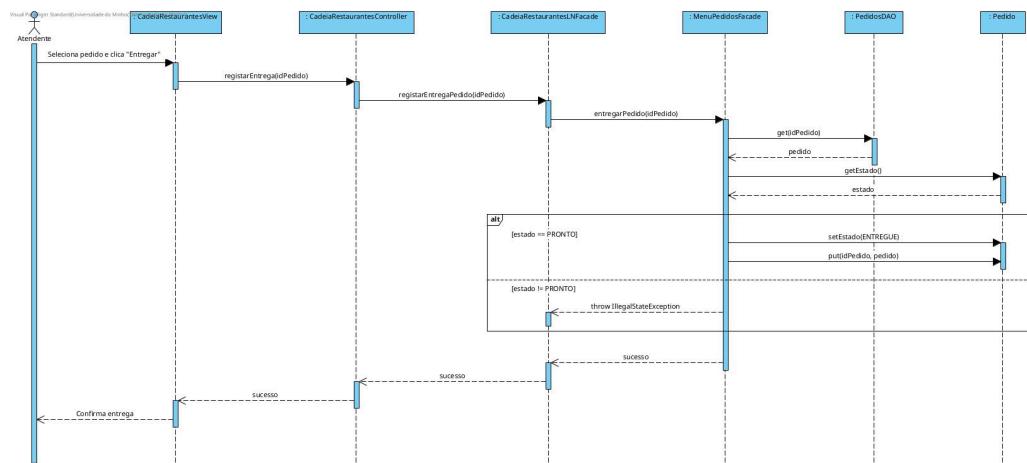


Figura 20: Diagrama de Sequência Implementado: **Registro da Entrega de um Pedido**.

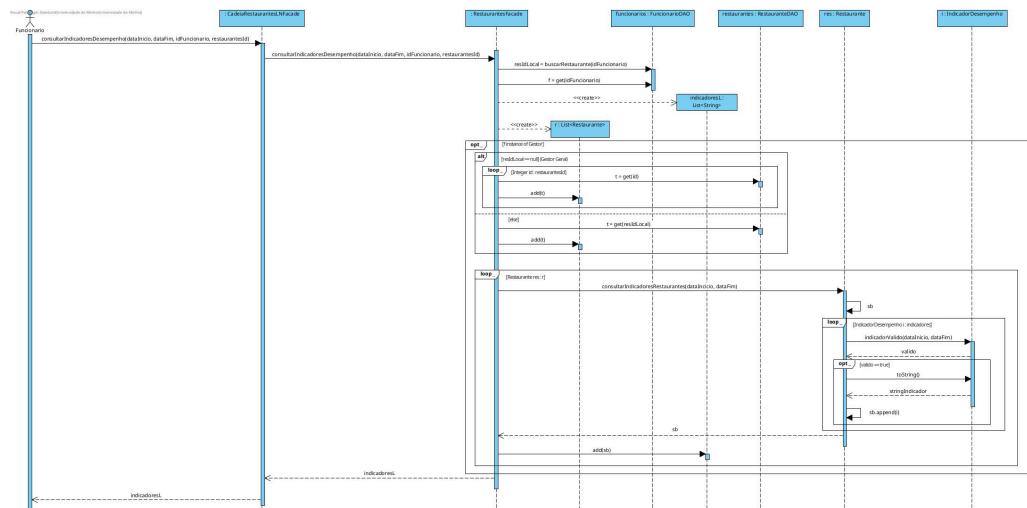


Figura 21: Diagrama de Sequência Implementado: **Consulta dos Indicadores de Desempenho**.

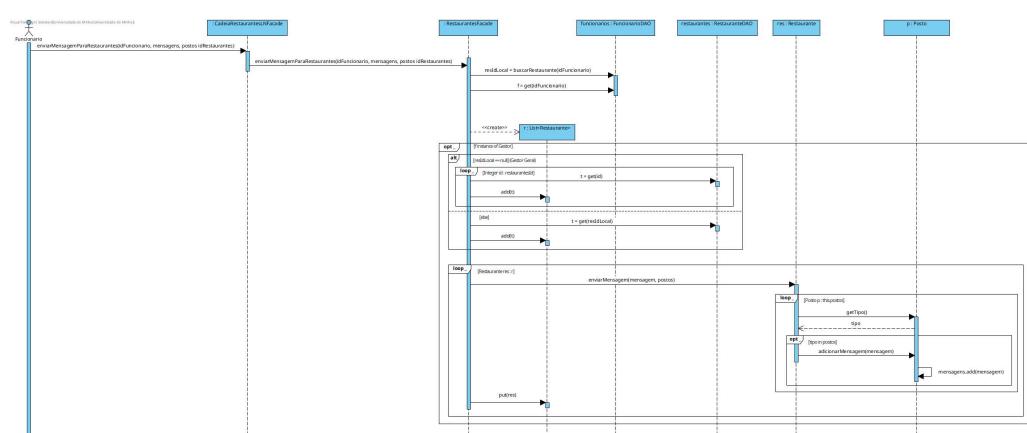


Figura 22: Diagrama de Sequência Implementado: **Envio de Mensagem aos Colaboradores**.

4.4. Diagrama de Packages

O **diagrama de packages** apresenta a organização lógica e a estrutura de pacotes Java do sistema, evidenciando uma hierarquia que reforça o **desacoplamento** entre a interface e os mecanismos de persistência:

- **dss.cadeiaRestaurantesUI**: Este pacote isola a **camada de apresentação**, comunicando com a lógica de negócio através da interface **ICadeiaRestaurantesLN**. Esta dependência aponta para a **CadeiaRestaurantesLNFacade**, que centraliza o acesso aos serviços do sistema;
- **dss.cadeiaRestaurantesLN**: O pacote central da aplicação está estruturado em dois grandes subsistemas que encapsulam a complexidade do negócio:
 - ▶ **SubSistemaMenuPedidos**: Agrupa a **MenuPedidosFacade** e as interfaces de gestão de artigos e vendas. Depende diretamente dos DAOs responsáveis por **Produtos**, **Pedidos**, **Menus** e **Ingredientes**;
 - ▶ **SubSistemaRestaurante**: Agrupa a **RestauranteFacade** (ou **RestaurantesFacade**) e gera as operações das unidades físicas. Depende dos DAOs de **Restaurante**, **IndicadorDesempenho** e **Funcionario**;
- **dss.cadeiaRestaurantesDL**: Contém a implementação dos **Data Access Objects**. De acordo com o diagrama, cada DAO gere internamente coleções do tipo **Map<Key, Value>** (ex: **Map<Integer, Pedido>**), servindo como uma **cache** em memória que facilita a manipulação de dados antes da sua persistência física no **MariaDB**.

As setas de dependência no diagrama confirmam um **fluxo de controlo descendente**, onde os pacotes de nível superior (UI) utilizam os de nível intermédio (LN), e estes, por sua vez, utilizam os de nível inferior (DL) para a gestão do estado persistente.

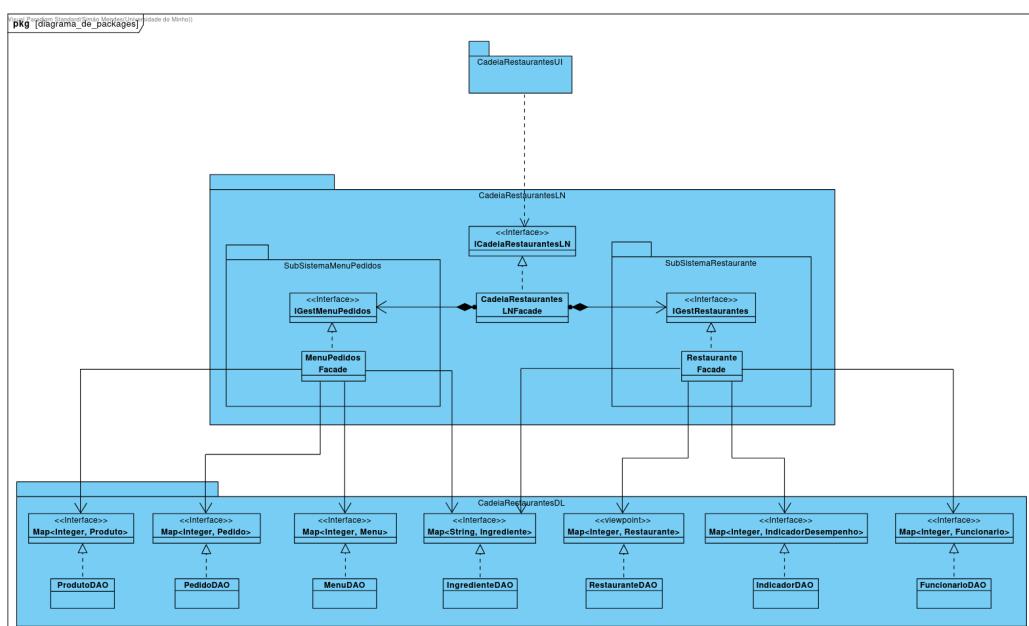


Figura 23: Organização Hierárquica de Pacotes Java e Dependências.

5. Manual de Utilização

O sistema opera através de uma **interface de linha de comandos** (CLI) organizada de forma hierárquica, garantindo uma navegação rápida e eficiente tanto para clientes como para colaboradores.

5.1. Requisitos de Sistema e Instalação

Para executar a aplicação de forma correta, devem ser assegurados os seguintes pré-requisitos técnicos:

- **Java Runtime (JRE/JDK):** Versão 21 ou superior para suporte às funcionalidades de *records* e *pattern matching*;
- **Servidor de Base de Dados:** MariaDB 11.x ativo na porta 3306;
- **Configuração:** Verifique o ficheiro DAOConfig.java para validar as credenciais de acesso local (URL, utilizador e *password*).

5.2. Configuração e Povoamento da Base de Dados

A **base de dados deve ser preparada** antes da execução através dos *scripts* localizados na pasta /mariadb:

1. **Criação:** Execute database_creation.sql para gerar a estrutura de tabelas e relações;
2. **Povoamento:** Execute populate_database.sql para inserir os dados base, como os Restaurantes, o catálogo de artigos e os perfis de Funcionario.

5.3. Guia de Navegação (Mapa de Menus)

A interface está estruturada em ramos distintos consoante o papel do utilizador:

- **Menu Inicial:** Ponto de entrada para seleção entre **Cliente** ou **Login de Funcionário**;
- **Fluxo de Cliente:**
 - Seleção de Restaurante → Menu de Artigos → Carrinho → Pagamento;
- **Fluxo de Funcionário (Cozinha):**
 - Lista de Pedidos → Seleção de Pedido → Gestão de Estado (Iniciar/Concluir);
- **Fluxo de Funcionário (Atendimento):**
 - Lista de Prontos → Confirmar Entrega.

- **Fluxo de Gestor:**

- **Dashboard de Indicadores:** Consulta de métricas com filtros por data e unidade;
- **Centro de Mensagens:** Sistema de *broadcast* para comunicação com os postos de trabalho.

5.4. Resolução de Problemas Comuns (*Troubleshooting*)

- **Falha na Autenticação:** Confirme que o ID utilizado existe na base de dados. IDs de teste padrão: 1 (Gestor), 5 (Cozinheiro), 11 (Atendente);
- **Erro de Conexão DB:** Garanta que o serviço MariaDB está em estado "Running" e que a porta 3306 não está bloqueada;
- **Caracteres Estranhos:** Caso utilize *Windows*, execute o comando chcp 65001 na consola antes de iniciar o programa para garantir suporte total a UTF-8.



Figura 24: Interface de Boas-vindas, Ponto de Entrada do Sistema e Início da Criação de um Pedido.

6. Conclusão

O desenvolvimento deste sistema de gestão para a cadeia de restaurantes permitiu consolidar a aplicação prática de conceitos avançados de **Engenharia de Software**. A solução final reflete um equilíbrio entre a **eficiência operacional** e a **robustez técnica**, o que garante que o fluxo de pedidos, desde a interação do cliente até à entrega final, seja processado de forma segura e transparente.

A adoção de uma **Arquitetura em Camadas** (Layered Architecture) revelou-se fundamental para assegurar o desacoplamento do sistema. A separação clara entre a **interface (UI)**, a **lógica de negócio (LN)** e a **persistência de dados (DL)** não só facilitou o **desenvolvimento paralelo**, como também garantiu que o sistema seja **facilmente escalável e de manutenção simplificada**. A implementação de padrões de desenho como o **DAO (Data Access Object)** e o **Facade** permitiu uma gestão eficiente da persistência em **MariaDB**, mantendo a integridade dos dados mesmo em cenários de utilização intensiva.

Em suma, o projeto **cumpre com sucesso os objetivos propostos** e entrega uma ferramenta que **automatiza processos críticos, otimiza a comunicação entre postos de trabalho** (Cozinha e Atendimento) e **fornecer aos gestores métricas fundamentais para a tomada de decisão**. A estrutura modular aqui apresentada serve como uma base sólida para futuras expansões, como a integração de novos métodos de pagamento ou a implementação de sistemas de fidelização de clientes.

7. Bibliografia

- [1] A. Dennis, B. H. Wixom, e D. Tegarden, *Systems Analysis & Design: An Object-Oriented Approach with UML*, 5.º ed. Wiley, 2015.
- [2] M. Blaha e J. Rumbaugh, *Object-Oriented Modeling and Design with UML*, 2.º ed. Prentice Hall, 2005.
- [3] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3.º ed. Addison-Wesley, 2004.
- [4] S. W. Ambler, *The Elements of UML 2.0 Style*. Cambridge University Press, 2005.
- [5] M. Nunes e H. O'Neill, *Fundamental de UML*, 3.º ed. FCA, 2007.
- [6] R. S. Wazlawick, *Análise e Design Orientados a Objetos para Sistemas de Informação*, 3.º ed. Grupo GEN, 2014.
- [7] J. C. Campos e A. N. Ribeiro, «Desenvolvimento de Sistemas de Software - Apresentação 2025/2026». [Em linha]. Disponível em: <http://www.di.uminho.pt/>
- [8] Oracle Corporation, «Java Platform, Standard Edition Documentation, Release 21». [Em linha]. Disponível em: <https://docs.oracle.com/en/java/javase/21/>
- [9] MariaDB Foundation, «MariaDB Server Documentation». [Em linha]. Disponível em: <https://mariadb.com/kb/en/documentation/>

8. Anexos

8.1. Script de Criação da Base de Dados

```
-- =====
-- SCRIPT 01: CRIAÇÃO DA BASE DE DADOS E ESTRUTURA
-- =====
-- Atualizado em: 26/12/2025
-- =====

DROP DATABASE IF EXISTS cadeiaRestaurantes;

CREATE DATABASE IF NOT EXISTS cadeiaRestaurantes;

USE cadeiaRestaurantes;

-- =====
-- UTILIZADOR
-- =====

DROP USER IF EXISTS 'funcionario'@'localhost';

CREATE USER IF NOT EXISTS 'funcionario'@'localhost' IDENTIFIED BY '';

GRANT ALL PRIVILEGES ON cadeiaRestaurantes.* TO 'funcionario'@'localhost';

FLUSH PRIVILEGES;

-- =====
-- TABELAS PRINCIPAIS
-- =====

CREATE TABLE IF NOT EXISTS restaurante (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    localizacao VARCHAR(200)
);

CREATE TABLE IF NOT EXISTS posto (
    id INT PRIMARY KEY AUTO_INCREMENT,
    tipo VARCHAR(50) NOT NULL,
    id_restaurante INT,
    FOREIGN KEY (id_restaurante) REFERENCES restaurante(id)
);

CREATE TABLE IF NOT EXISTS mensagem (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_posto INT NOT NULL,
```

```

        texto TEXT NOT NULL,
        data_hora TIMESTAMP DEFAULT NOW(),
        FOREIGN KEY (id_posto) REFERENCES posto(id)
    );

CREATE TABLE IF NOT EXISTS ingrediente (
    nome VARCHAR(50) NOT NULL PRIMARY KEY,
    preco DOUBLE NOT NULL
);

CREATE TABLE IF NOT EXISTS stock (
    id_restaurante INT,
    nome_ingrediente VARCHAR(50),
    quantidade INT NOT NULL,
    PRIMARY KEY (id_restaurante, nome_ingrediente),
    FOREIGN KEY (id_restaurante) REFERENCES restaurante(id),
    FOREIGN KEY (nome_ingrediente) REFERENCES ingrediente(nome)
);

-- =====
-- FUNCIONÁRIOS
-- =====

CREATE TABLE IF NOT EXISTS funcionario (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    nif INT NOT NULL,
    id_posto INT,
    FOREIGN KEY (id_posto) REFERENCES posto(id)
);

CREATE TABLE IF NOT EXISTS atendente (
    id INT PRIMARY KEY,
    FOREIGN KEY (id) REFERENCES funcionario(id)
);

CREATE TABLE IF NOT EXISTS cozinheiro (
    id INT PRIMARY KEY,
    FOREIGN KEY (id) REFERENCES funcionario(id)
);

CREATE TABLE IF NOT EXISTS gestor (
    id INT PRIMARY KEY,
    FOREIGN KEY (id) REFERENCES funcionario(id)
);

-- =====
-- INDICADORES
-- =====

CREATE TABLE IF NOT EXISTS indicador (
    id INT PRIMARY KEY AUTO_INCREMENT,

```

```

    data DATE NOT NULL,
    faturacaoTotal DOUBLE DEFAULT 0,
    totalPedidos INT DEFAULT 0,
    tempoMedioPreparacao INT DEFAULT 0,
    id_restaurante INT NOT NULL,
    FOREIGN KEY (id_restaurante) REFERENCES restaurante(id) ON DELETE CASCADE
);

-- =====
-- PRODUTOS E MENUS
-- =====

CREATE TABLE IF NOT EXISTS produto (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    preco DECIMAL(10,2) NOT NULL,
    tamanho VARCHAR(20)
);

CREATE TABLE IF NOT EXISTS produto_ingredientes (
    id_produto INT NOT NULL,
    nome_ingrediente VARCHAR(50) NOT NULL,
    PRIMARY KEY (id_produto, nome_ingrediente),
    FOREIGN KEY (id_produto) REFERENCES produto(id) ON DELETE CASCADE,
    FOREIGN KEY (nome_ingrediente) REFERENCES ingrediente(nome) ON DELETE
CASCADE
);

CREATE TABLE IF NOT EXISTS menu (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    tipo VARCHAR(20) NOT NULL,
    preco DECIMAL(10,2),
    categoria VARCHAR(50)
);

CREATE TABLE IF NOT EXISTS menu_produtos (
    id_menu INT NOT NULL,
    id_produto INT NOT NULL,
    PRIMARY KEY (id_menu, id_produto),
    FOREIGN KEY (id_menu) REFERENCES menu(id) ON DELETE CASCADE,
    FOREIGN KEY (id_produto) REFERENCES produto(id) ON DELETE CASCADE
);

-- =====
-- PEDIDOS (Alinhado com PedidoDAO.java)
-- =====

CREATE TABLE IF NOT EXISTS pedido (
    id INT PRIMARY KEY AUTO_INCREMENT,
    estado VARCHAR(50) NOT NULL,
    tipo_consumo VARCHAR(50) NOT NULL,

```

```

        id_restaurante INT NOT NULL,
        valor_total DECIMAL(10,2) DEFAULT 0.0,
        FOREIGN KEY (id_restaurante) REFERENCES restaurante(id)
    );

CREATE TABLE IF NOT EXISTS linha_pedido (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_pedido INT NOT NULL,
    quantidade INT NOT NULL,
    preco_venda DECIMAL(10,2) NOT NULL,
    id_artigo VARCHAR(50) NOT NULL,
    tipo_artigo VARCHAR(20) NOT NULL,
    nota TEXT,
    ingredientes_personalizados TEXT,
    FOREIGN KEY (id_pedido) REFERENCES pedido(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS pagamento (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_pedido INT NOT NULL UNIQUE,
    valor DECIMAL(10,2) NOT NULL,
    metodo VARCHAR(30) NOT NULL,
    detalhes VARCHAR(255),
    FOREIGN KEY (id_pedido) REFERENCES pedido(id) ON DELETE CASCADE
);

-- =====
-- FIM DO SCRIPT
-- =====

```

8.2. Script de População da Base de Dados

```

-- =====
-- SCRIPT 02: POVOAMENTO DA BASE DE DADOS
-- =====
-- Atualizado em: 26/12/2025
-- Dados de teste para demonstração do sistema
-- =====

USE cadeiaRestaurantes;

-- =====
-- 1. LIMPEZA
-- =====

DELETE FROM pagamento;
DELETE FROM linha_pedido;
DELETE FROM pedido;
DELETE FROM mensagem;
DELETE FROM indicador;

```

```

DELETE FROM atendente;
DELETE FROM cozinhiero;
DELETE FROM gestor;
DELETE FROM funcionario;
DELETE FROM stock;
DELETE FROM posto;
DELETE FROM restaurante;
DELETE FROM menu_produtos;
DELETE FROM menu;
DELETE FROM produto_ingredientes;
DELETE FROM produto;
DELETE FROM ingrediente;

-- =====
-- 2. RESET AUTO_INCREMENT
-- =====

ALTER TABLE mensagem AUTO_INCREMENT = 1;
ALTER TABLE indicador AUTO_INCREMENT = 1;
ALTER TABLE funcionario AUTO_INCREMENT = 1;
ALTER TABLE posto AUTO_INCREMENT = 1;
ALTER TABLE restaurante AUTO_INCREMENT = 1;
ALTER TABLE produto AUTO_INCREMENT = 1;
ALTER TABLE menu AUTO_INCREMENT = 1;
ALTER TABLE linha_pedido AUTO_INCREMENT = 1;
ALTER TABLE pedido AUTO_INCREMENT = 1;
ALTER TABLE pagamento AUTO_INCREMENT = 1;

-- =====
-- 3. RESTAURANTES
-- =====

INSERT INTO restaurante (nome, localizacao) VALUES
('Restaurante Centro', 'Centro da cidade'),
('Restaurante Norte', 'Zona Norte'),
('Restaurante Sul', 'Zona Sul');

-- =====
-- 4. POSTOS DE TRABALHO
-- =====

INSERT INTO posto (tipo, id_restaurante) VALUES
('BALCAO', 1),
('COZINHA', 1),
('GESTAO', 1),
('BALCAO', 2),
('COZINHA', 2),
('GESTAO', 2),
('BALCAO', 3),
('COZINHA', 3),
('GESTAO', 3);

```

```

-- =====
-- 5. FUNCIONÁRIOS
-- =====

INSERT INTO funcionario (nome, nif, id_posto) VALUES
-- Gestor Geral (sem posto específico)
('Admin Geral', 999999999, NULL),
-- Gestores dos Restaurantes
('João Gestor Centro', 111111111, 3),
('Maria Gestora Norte', 222222222, 6),
('Carlos Gestor Sul', 333333333, 9),
-- Cozinheiros
('António Cozinha Centro', 444444444, 2),
('Beatriz Chef Centro', 555555555, 2),
('Duarte Cozinha Norte', 666666666, 5),
('Eva Chef Norte', 777777777, 5),
('Francisco Cozinha Sul', 888888888, 8),
('Gabriela Chef Sul', 123456789, 8),
-- Atendentes
('Helena Atendente Centro', 234567890, 1),
('Ivo Atendente Centro', 345678901, 1),
('Joana Atendente Norte', 456789012, 4),
('Kiko Atendente Norte', 567890123, 4),
('Luísa Atendente Sul', 678901234, 7),
('Miguel Atendente Sul', 789012345, 7);

-- Tipos de Funcionários
INSERT INTO gestor (id) VALUES (1), (2), (3), (4);
INSERT INTO cozinheiro (id) VALUES (5), (6), (7), (8), (9), (10);
INSERT INTO atendente (id) VALUES (11), (12), (13), (14), (15), (16);

-- =====
-- 6. INGREDIENTES
-- =====

INSERT INTO ingrediente (nome, preco) VALUES
('Tomate', 0.50),
('Cebola', 0.30),
('Alho', 0.20),
('Azeite', 0.50),
('Sal', 0.10),
('Pimenta', 0.15),
('Arroz', 1.00),
('Feijão', 1.20),
('Carne Bovina', 2.00),
('Frango', 1.50),
('Peixe', 2.50),
('Batata', 0.40),
('Cenoura', 0.35),
('Alface', 0.40),
('Queijo', 0.70),
('Pão', 0.50),

```

```

('Pão Hambúrguer', 0.80),
('Bacon', 1.00),
('Ovo', 0.60);

-- =====
-- 7. STOCK DOS RESTAURANTES
-- =====

INSERT INTO stock (id_restaurante, nome_ingrediente, quantidade) VALUES
-- Restaurante Centro
(1, 'Tomate', 50),
(1, 'Cebola', 30),
(1, 'Alho', 20),
(1, 'Azeite', 10),
(1, 'Carne Bovina', 25),
(1, 'Frango', 20),
(1, 'Pão Hambúrguer', 100),
(1, 'Alface', 40),
(1, 'Queijo', 30),
(1, 'Bacon', 25),
-- Restaurante Norte
(2, 'Tomate', 40),
(2, 'Cebola', 25),
(2, 'Alho', 15),
(2, 'Azeite', 8),
(2, 'Peixe', 18),
(2, 'Batata', 35),
(2, 'Frango', 15),
(2, 'Queijo', 20),
-- Restaurante Sul
(3, 'Tomate', 35),
(3, 'Cebola', 20),
(3, 'Alho', 12),
(3, 'Azeite', 6),
(3, 'Frango', 15),
(3, 'Queijo', 20),
(3, 'Batata', 30),
(3, 'Arroz', 25);

-- =====
-- 8. INDICADORES DE DESEMPENHO
-- =====

INSERT INTO indicador (data, faturacaoTotal, totalPedidos,
tempoMedioPreparacao, id_restaurante) VALUES
-- Restaurante Centro (últimos 30 dias)
(CURDATE() - INTERVAL 30 DAY, 1250.50, 45, 12, 1),
(CURDATE() - INTERVAL 25 DAY, 980.75, 38, 15, 1),
(CURDATE() - INTERVAL 20 DAY, 2100.00, 65, 10, 1),
(CURDATE() - INTERVAL 15 DAY, 1750.25, 52, 14, 1),
(CURDATE() - INTERVAL 10 DAY, 890.30, 32, 16, 1),
(CURDATE() - INTERVAL 5 DAY, 1450.60, 48, 11, 1),

```

```

(CURDATE() - INTERVAL 1 DAY, 1650.40, 55, 13, 1),
-- Restaurante Norte (últimos 30 dias)
(CURDATE() - INTERVAL 30 DAY, 980.25, 32, 18, 2),
(CURDATE() - INTERVAL 25 DAY, 750.50, 28, 20, 2),
(CURDATE() - INTERVAL 20 DAY, 1650.75, 52, 15, 2),
(CURDATE() - INTERVAL 15 DAY, 1200.30, 42, 17, 2),
(CURDATE() - INTERVAL 10 DAY, 850.60, 30, 19, 2),
(CURDATE() - INTERVAL 5 DAY, 1100.40, 38, 16, 2),
(CURDATE() - INTERVAL 1 DAY, 1350.20, 45, 14, 2),
-- Restaurante Sul (últimos 30 dias)
(CURDATE() - INTERVAL 30 DAY, 650.75, 25, 22, 3),
(CURDATE() - INTERVAL 25 DAY, 820.30, 30, 19, 3),
(CURDATE() - INTERVAL 20 DAY, 1450.60, 48, 16, 3),
(CURDATE() - INTERVAL 15 DAY, 1100.25, 40, 18, 3),
(CURDATE() - INTERVAL 10 DAY, 720.50, 26, 21, 3),
(CURDATE() - INTERVAL 5 DAY, 950.75, 35, 17, 3),
(CURDATE() - INTERVAL 1 DAY, 1250.40, 42, 15, 3);

-- =====
-- 9. PRODUTOS
-- =====

-- Inserir produtos
INSERT INTO produto (id, nome, preco, tamanho) VALUES
(1, 'Hambúrguer Clássico', 6.50, 'M'),
(2, 'Cheeseburger Duplo', 8.50, 'L'),
(3, 'Bitoque no Prato', 10.00, 'M'),
(4, 'Salada de Frango', 7.50, 'M'),
(5, 'Batatas Fritas', 2.50, 'M'),
(6, 'Refrigerante', 2.00, 'M'),
(7, 'Água 50cl', 1.50, 'M'),
(8, 'Sundae Baunilha', 3.00, 'M');

-- Associar ingredientes aos produtos
INSERT INTO produto_ingredientes (id_produto, nome_ingrediente) VALUES
-- Hambúrguer Clássico (ID: 1)
(1, 'Pão Hambúrguer'),
(1, 'Alface'),
(1, 'Carne Bovina'),
(1, 'Tomate'),
-- Cheeseburger Duplo (ID: 2)
(2, 'Pão Hambúrguer'),
(2, 'Carne Bovina'),
(2, 'Queijo'),
(2, 'Bacon'),
-- Bitoque (ID: 3)
(3, 'Arroz'),
(3, 'Batata'),
(3, 'Carne Bovina'),
(3, 'Ovo'),
-- Salada de Frango (ID: 4)

```

```

(4, 'Alface'),
(4, 'Tomate'),
(4, 'Cenoura'),
(4, 'Frango');

-- =====
-- 10. MENUS
-- =====

-- Inserir menus
INSERT INTO menu (id, nome, tipo, preco, categoria) VALUES
(9, 'Menu Estudante', 'PACK', 8.00, NULL),
(10, 'Menu Familiar', 'PACK', 25.00, NULL);

-- Associar produtos aos menus
INSERT INTO menu_produtos (id_menu, id_produto) VALUES
-- Menu Estudante (ID: 9)
(9, 1), -- Hambúrguer
(9, 5), -- Batatas
(9, 6), -- Refrigerante
-- Menu Familiar (ID: 10)
(10, 1), -- Hambúrguer
(10, 2), -- Cheeseburger
(10, 5), -- Batatas
(10, 6); -- Refrigerante

-- =====
-- 11. PEDIDOS DE TESTE
-- =====

-- RESTAURANTE CENTRO (ID: 1)
INSERT INTO pedido (estado, tipo_consumo, id_restaurante, valor_total)
VALUES
('REGISTADO', 'LOCAL', 1, 25.50),
('REGISTADO', 'TAKE_AWAY', 1, 12.00),
('EM_PREPARACAO', 'LOCAL', 1, 45.00),
('PRONTO', 'TAKE_AWAY', 1, 18.50),
('PRONTO', 'LOCAL', 1, 32.00),
('ENTREGUE', 'LOCAL', 1, 50.00);

-- RESTAURANTE NORTE (ID: 2)
INSERT INTO pedido (estado, tipo_consumo, id_restaurante, valor_total)
VALUES
('REGISTADO', 'LOCAL', 2, 15.00),
('PRONTO', 'TAKE_AWAY', 2, 22.00);

-- RESTAURANTE SUL (ID: 3)
INSERT INTO pedido (estado, tipo_consumo, id_restaurante, valor_total)
VALUES
('REGISTADO', 'LOCAL', 3, 10.00);

```

```
-- =====
-- 12. MENSAGENS DE TESTE (OPCIONAL)
-- =====

INSERT INTO mensagem (id_posto, texto) VALUES
(2, 'Atenção: Novo fornecedor de ingredientes a partir de segunda-feira'),
(1, 'Lembrete: Verificar stock de embalagens take-away'),
(3, 'Reunião de equipa agendada para sexta às 15h');

-- =====
-- FIM DO SCRIPT
-- =====
```

8.3. *Script* de Visulização da Base de Dados

```
-- =====
-- SCRIPT 03: VISUALIZAÇÃO E VERIFICAÇÃO DE DADOS
-- =====
-- Atualizado em: 26/12/2025
-- Queries para validar a estrutura e dados da BD
-- =====

USE cadeiaRestaurantes;

-- =====
-- 1. RESTAURANTES E ESTRUTURA ORGANIZACIONAL
-- =====

SELECT '--- RESTAURANTES ---' AS '';
SELECT id, nome, localizacao
FROM restaurante
ORDER BY id;

SELECT '--- POSTOS DE TRABALHO ---' AS '';
SELECT p.id, p.tipo, p.id_restaurante, r.nome AS restaurante
FROM posto p
LEFT JOIN restaurante r ON p.id_restaurante = r.id
ORDER BY p.id_restaurante, p.tipo;

-- =====
-- 2. FUNCIONÁRIOS
-- =====

SELECT '--- TODOS OS FUNCIONÁRIOS ---' AS '';
SELECT f.id, f.nome, f.nif, p.tipo AS posto, r.nome AS restaurante
FROM funcionario f
LEFT JOIN posto p ON f.id_posto = p.id
LEFT JOIN restaurante r ON p.id_restaurante = r.id
ORDER BY f.id;
```

```

SELECT '==== GESTORES ===' AS '';
SELECT g.id, f.nome, f.nif,
       COALESCE(r.nome, 'Gestor Geral') AS restaurante
FROM gestor g
JOIN funcionario f ON g.id = f.id
LEFT JOIN posto p ON f.id_posto = p.id
LEFT JOIN restaurante r ON p.id_restaurante = r.id
ORDER BY g.id;

SELECT '==== COZINHEIROS ===' AS '';
SELECT c.id, f.nome, f.nif, r.nome AS restaurante
FROM cozinheiro c
JOIN funcionario f ON c.id = f.id
LEFT JOIN posto p ON f.id_posto = p.id
LEFT JOIN restaurante r ON p.id_restaurante = r.id
ORDER BY c.id;

SELECT '==== ATENDENTES ===' AS '';
SELECT a.id, f.nome, f.nif, r.nome AS restaurante
FROM atendente a
JOIN funcionario f ON a.id = f.id
LEFT JOIN posto p ON f.id_posto = p.id
LEFT JOIN restaurante r ON p.id_restaurante = r.id
ORDER BY a.id;

-- =====
-- 3. INGREDIENTES E STOCK
-- =====

SELECT '==== INGREDIENTES (CATÁLOGO) ===' AS '';
SELECT nome, CONCAT(preco, '€') AS preco
FROM ingrediente
ORDER BY nome;

SELECT '==== STOCK POR RESTAURANTE ===' AS '';
SELECT r.nome AS restaurante,
       s.nome_ingrediente,
       s.quantidade,
       i.preco AS preco_unitario
FROM stock s
JOIN restaurante r ON s.id_restaurante = r.id
JOIN ingrediente i ON s.nome_ingrediente = i.nome
ORDER BY s.id_restaurante, s.nome_ingrediente;

-- =====
-- 4. INDICADORES DE DESEMPENHO
-- =====

SELECT '==== INDICADORES (ÚLTIMOS 30 DIAS) ===' AS '';
SELECT r.nome AS restaurante,
       i.data,
       CONCAT(i.faturacaoTotal, '€') AS faturacao,

```

```

        i.totalPedidos AS pedidos,
        CONCAT(i.tempoMedioPreparacao, ' min') AS tempo_medio
    FROM indicador i
    JOIN restaurante r ON i.id_restaurante = r.id
    WHERE i.data >= CURDATE() - INTERVAL 30 DAY
    ORDER BY i.id_restaurante, i.data DESC;

SELECT '--- RESUMO INDICADORES POR RESTAURANTE ---' AS '';
SELECT r.nome AS restaurante,
       COUNT(*) AS total_indicadores,
       CONCAT(SUM(i.faturacaoTotal), '€') AS faturacao_total,
       SUM(i.totalPedidos) AS total_pedidos,
       CONCAT(ROUND(AVG(i.tempoMedioPreparacao), 1), ' min') AS
tempo_medio_geral
    FROM indicador i
    JOIN restaurante r ON i.id_restaurante = r.id
   GROUP BY r.id, r.nome
  ORDER BY r.id;

-- =====
-- 5. PEDIDOS
-- =====

SELECT '--- PEDIDOS (TODOS) ---' AS '';
SELECT p.id,
       r.nome AS restaurante,
       p.estado,
       p.tipo_consumo,
       CONCAT(p.valor_total, '€') AS valor
    FROM pedido p
    JOIN restaurante r ON p.id_restaurante = r.id
   ORDER BY p.id;

SELECT '--- PEDIDOS POR ESTADO ---' AS '';
SELECT r.nome AS restaurante,
       p.estado,
       COUNT(*) AS quantidade,
       CONCAT(SUM(p.valor_total), '€') AS valor_total
    FROM pedido p
    JOIN restaurante r ON p.id_restaurante = r.id
   GROUP BY r.id, r.nome, p.estado
  ORDER BY r.id, p.estado;

-- =====
-- 6. LINHAS DE PEDIDO
-- =====

SELECT '--- LINHAS DE PEDIDO ---' AS '';
SELECT lp.id,
       lp.id_pedido,
       lp.quantidade,
       CONCAT(lp.preco_venda, '€') AS preco,

```

```

lp.tipo_artigo,
lp.id_artigo,
LEFT(COALESCE(lp.nota, '-'), 30) AS nota,
CASE
    WHEN lp.ingredientes_personalizados IS NOT NULL
    THEN 'SIM'
    ELSE 'NÃO'
END AS personalizado
FROM linha_pedido lp
ORDER BY lp.id_pedido, lp.id;

SELECT '== LINHAS COM INGREDIENTES PERSONALIZADOS ==' AS '';
SELECT lp.id,
lp.id_pedido,
lp.quantidade,
lp.tipo_artigo,
lp.id_artigo,
lp.ingredientes_personalizados
FROM linha_pedido lp
WHERE lp.ingredientes_personalizados IS NOT NULL
ORDER BY lp.id_pedido;

-- =====
-- 7. PAGAMENTOS
-- =====

SELECT '== PAGAMENTOS ==' AS '';
SELECT pg.id,
pg.id_pedido,
CONCAT(pg.valor, '€') AS valor,
pg.metodo,
pg.detalhes
FROM pagamento pg
ORDER BY pg.id_pedido;

-- =====
-- 8. MENSAGENS
-- =====

SELECT '== MENSAGENS (ÚLTIMAS 10) ==' AS '';
SELECT m.id,
p.tipo AS posto,
r.nome AS restaurante,
LEFT(m.texto, 50) AS mensagem,
m.data_hora
FROM mensagem m
JOIN posto p ON m.id_posto = p.id
LEFT JOIN restaurante r ON p.id_restaurante = r.id
ORDER BY m.data_hora DESC
LIMIT 10;

```

```

-- 9. PRODUTOS E MENUS
-- =====

SELECT '==== PRODUTOS ===' AS '';
SELECT * FROM produto LIMIT 10;

SELECT '==== MENUS ===' AS '';
SELECT * FROM menu LIMIT 10;

-- =====
-- 10. ESTATÍSTICAS GERAIS
-- =====

SELECT '==== ESTATÍSTICAS GERAIS ===' AS '';

SELECT 'Total Restaurantes' AS metrica, COUNT(*) AS valor FROM restaurante
UNION ALL
SELECT 'Total Funcionários', COUNT(*) FROM funcionario
UNION ALL
SELECT 'Total Gestores', COUNT(*) FROM gestor
UNION ALL
SELECT 'Total Cozinheiros', COUNT(*) FROM cozinheiro
UNION ALL
SELECT 'Total Atendentes', COUNT(*) FROM atendente
UNION ALL
SELECT 'Total Ingredientes', COUNT(*) FROM ingrediente
UNION ALL
SELECT 'Total Pedidos', COUNT(*) FROM pedido
UNION ALL
SELECT 'Pedidos Registados', COUNT(*) FROM pedido WHERE estado = 'REGISTADO'
UNION ALL
SELECT 'Pedidos Em Preparação', COUNT(*) FROM pedido WHERE estado =
'EM_PREPARACAO'
UNION ALL
SELECT 'Pedidos Prontos', COUNT(*) FROM pedido WHERE estado = 'PRONTO'
UNION ALL
SELECT 'Pedidos Entregues', COUNT(*) FROM pedido WHERE estado = 'ENTREGUE';

-- =====
-- FIM DO SCRIPT
-- =====

```

8.4. Enunciado do Projeto

Desenvolvimento de Sistemas de Software

Licenciatura em Engenharia Informática

Departamento de Informática

Universidade do Minho

2025/2026

Enunciado do Trabalho

António Nestor Ribeiro, Tiago Oliveira, Afonso Sousa

(disponibilizado em 26/09/2025)

Conteúdo

1	Introdução	1
2	Objectivo do trabalho	1
2.1	O pedido	2
2.2	Informação para a gestão	2
3	Realização do trabalho	2
3.1	Entrega intermédia	3
3.2	Entrega final	3
4	Apresentação e discussão do trabalho	4
5	Avaliação	4
6	Grupos de Trabalho	5

1 Introdução

Este documento apresenta o enunciado do trabalho prático da Unidade Curricular (UC) de Desenvolvimento de Sistemas Software para o ano lectivo 2025/2026. **Leia-o com atenção**, já que descreve, não só o sistema a desenvolver, como o processo que deve seguir para a realização do trabalho. Quaisquer dúvidas devem ser esclarecidas junto dos docentes da UC.

2 Objectivo do trabalho

Uma cadeia de restaurantes de fast-food pretende ter um sistema integrado que permita automatizar todos os aspectos do seu funcionamento nos restaurantes, desde os pedidos dos clientes até ao processo de elaboração dos pratos e à entrega dos mesmos aos clientes. Pretende-se também obter informação de funcionamento para a gestão da cadeia de restaurantes, no que concerne a volume de pedidos, stocks dos produtos, tempo médio de espera por uma refeição, entre outros.

Os restaurantes têm todos o mesmo modelo de funcionamento: os clientes escolhem o que pretendem consumir em ecrãs tácteis que estão à entrada do espaço e após finalizarem de compor o pedido, este é passado para a produção e empurramento (ou embalamento no caso de ser para take away). A confecção da refeição passa por diversos estágios e tem diversos empregados que assumem funções diferentes: um grelha a carne, outro frita as batatas, outro trata dos ovos, outro dos legumes, etc. Nem todos os pratos precisam de passar por todos estes estágios e é objectivo do sistema a construir que as tarefas estejam devidamente organizadas para que as refeições não fiquem com alimentos à espera da confeção de outros. Por exemplo, não faz sentido para um pedido de um prego no pão com ovo, ter o bife grelhado e o pão aquecido, mas ficar à espera de que o ovo seja estrelado.

Cada posto de um funcionário terá um display em que sabe o que tem de fazer e tem uma visão dos pedidos que se seguem e de algumas notas que tenham sido enviadas pelo cliente na altura em que fez o pedido. Poderá também reordenar os pedidos caso exista algo imprevisto (ex: falta momentânea de um ingrediente que terá de ser trazido do armazém), sendo que tal poderá ter impacto nas tarefas de outros funcionários.

Existem também funcionários que estão encarregados de efectuar a entrega dos pratos aos clientes que fazem a refeição dentro do restaurante ou então de embalar os pratos para os recipientes de take-away. Em ambas as situações, todos os componentes de um pedido deverão estar confeccionados na altura da entrega ao cliente.

2.1 O pedido

Quando o cliente faz o pedido nos ecrãs que existem para o efeito no restaurante, escolhe de uma série de propostas previamente existentes. Algumas dessas propostas admitem que se possam acrescentar ou retirar ingredientes. Em função dos ingredientes que são adicionados, é possível que surjam opções relativas à sua confecção e seja necessário perguntar ao cliente. Algumas das propostas aparecem sob a forma de um menu já composto. Mesmo nessas circunstâncias é possível escolher várias opções para os diferentes elementos do menu (por exemplo, na bebida podemos ter a opção de água, refrigerante, limonada, etc.).

Alguns alimentos podem ter a indicação de alergénios e o cliente pode decidir que não pretende ter algum deles ou a sua totalidade. Nessas circunstâncias a aplicação deve validar as propostas de refeição e os menus que ainda é possível oferecer sem que esses alergénios estejam presentes. Poderá também ser necessário alterar o processo de confecção, no sentido de pedir aos postos que não incluam os referidos alergénios na execução do pedido.

2.2 Informação para a gestão

O objectivo da construção de sistema integrado atrás referido deriva do facto de que a gestão pretende ter indicadores sobre os pedidos que os clientes fazem, sobre os produtos/refeições mais vendidos e poder controlar as necessidades de stock de produtos e eventuais necessidades de encomendar esses mesmos produtos. Pretende-se também ter indicadores sobre o tempo médio de atendimento dos pedidos e sobre as funções mais requisitadas na elaboração dos pedidos, para eventualmente abrir mais postos daquele tipo. O sistema deverá ter um conjunto de funcionalidades que permita à gestão da cadeia de restaurante obter este tipo de informação.

Para esclarecimento de eventuais dúvidas, contacte os docentes de DSS.

3 Realização do trabalho

A concepção e desenvolvimento da aplicação deverá seguir uma abordagem baseada em modelos (suportada por UML), de acordo com o processo de entregas faseadas descrito nas aulas teóricas. A aplicação deverá ser desenvolvida utilizando uma arquitectura multi-camada e tecnologias orientadas a objectos (preferencialmente, Java). Irá ser criado um repositório no GitHub¹ para cada grupo, onde deverá ser mantida a versão actualizada do trabalho.

¹<https://github.com>

Para facilitar o processo de concepção e desenvolvimento, o trabalho será realizado em duas fases.

3.1 Entrega intermédia

Análise de requisitos – a entregar até às 23h59 de 19 de outubro.

Objectivos:

- Um Modelo de Domínio com as entidades relevantes
- Um Modelo de Use Case (diagramas mais especificações do Use Case) com as funcionalidades propostas para o sistema

O resultado desta fase será sujeito a avaliação qualitativa.

3.2 Entrega final

Modelação conceptual e implementação da solução – a entregar até às 23h59 de 9 de janeiro.

Objectivos:

- Uma arquitectura conceptual do sistema, capaz de suportar os requisitos identificados.
- Os modelos comportamentais necessários para descrever o comportamento pretendido para o sistema
- Os modelos que considere necessários à descrição da implementação do sistema
- A implementação do sistema
- Documento técnico com todos os modelos desenvolvidos (em PDF).

Pretende-se que o documento técnico sirva de apoio à análise do trabalho, pelo que **deverá ter a seguinte estrutura:**

- **Capa com identificação** da Unidade Curricular, **do grupo (com fotos dos elementos)** e o URL do **repositório do trabalho**.
- Descrição dos resultados obtidos (máximo uma página).
- Diagramas relativos à **análise de requisitos** (Modelação de Domínio, Diagramas de Casos de Uso e correspondentes descrições dos casos de uso).

- Diagramas relativos à **modelação conceptual da solução** proposta (Diagramas de Classe e de Sequência).
- Diagramas com a descrição da **solução efectivamente implementada** (Diagramas de Classe, de Sequência, de Componentes e de *packages*).
- Manual de utilização do sistema desenvolvido.
- Em anexo, este enunciado.

Os diagramas mencionados acima podem ser complementados com outros que considerem relevante incluir.

4 Apresentação e discussão do trabalho

Para a apresentação do trabalho deverão preparar uma apresentação com a duração máxima de 15 minutos. Esta apresentação deverá descrever a solução e a abordagem seguida para a atingir, desde a análise dos cenários até a implementação e demonstração da solução final. A apresentação deverá terminar com uma análise crítica dos resultados obtidos.

Após essa apresentação, seguir-se-á um período de análise e discussão do trabalho de até 30 minutos.

5 Avaliação

A apresentação e discussão final do trabalho será realizada na semana de 12 de janeiro de 2026, em horários a combinar. A **presença** na discussão do trabalho é **obrigatória**.

Os pesos relativos de cada componente do trabalho serão os seguintes:

- Modelo de domínio e análise de requisitos: 25%
- Modelação conceptual: 25%
- Modelação final e implementação: 35%
- Apresentação e discussão: 15%

A nota de cada elemento do grupo será individual, tendo em consideração a nota do trabalho e a avaliação por pares. A equipa docente reserva-se a possibilidade de ajustar as notas, em função da sua avaliação de cada elemento durante a discussão do trabalho.

6 Grupos de Trabalho

Os grupos de trabalho deverão obrigatoriamente ser constituídos por de 3 a 5 elementos. A definição dos grupos de trabalho será realizada no Blackboard, **terminando a 8 de outubro.**