Restaurant Monsieur José

Davi Brasileiro Gomes - 241020741
Davi Galvão Guerra - 241038577
Roberto Ribeiro Correa de Oliveira Andrade Neto - 242009936
Thiago Fernandes Carvalho de Souza - 221030885

Dep. Ciência da Computação – Universidade de Brasília (UnB)
CIC0197 - Técnicas de Programação 1

1. Descrição do Problema

Monsieur José era um senhor francês apaixonado por gastronomia. Ao se mudar para o Brasil, encantou-se pela cultura local e decidiu abrir um pequeno restaurante em uma esquina tranquila, onde misturava técnicas francesas com ingredientes brasileiros. O restaurante, batizado de Monsieur José, rapidamente conquistou o paladar dos moradores com seu charme rústico e pratos autênticos.

Com o passar dos meses, a fama do local se espalhou. Filas se formavam diariamente e as reservas se tornavam disputadas. Para acompanhar esse crescimento exponencial, José decidiu investir em tecnologia e encomendou a criação de um site moderno. O objetivo era facilitar a organização dos pedidos para entrega e o gerenciamento das reservas, garantindo que a experiência dos clientes fosse tão refinada quanto seus pratos, além de disponibilizar uma seção reservada ao gerente do restaurante, concentrando todas as informações relevantes do estabelecimento de maneira acessível e organizada.

2. Regras de Negócio

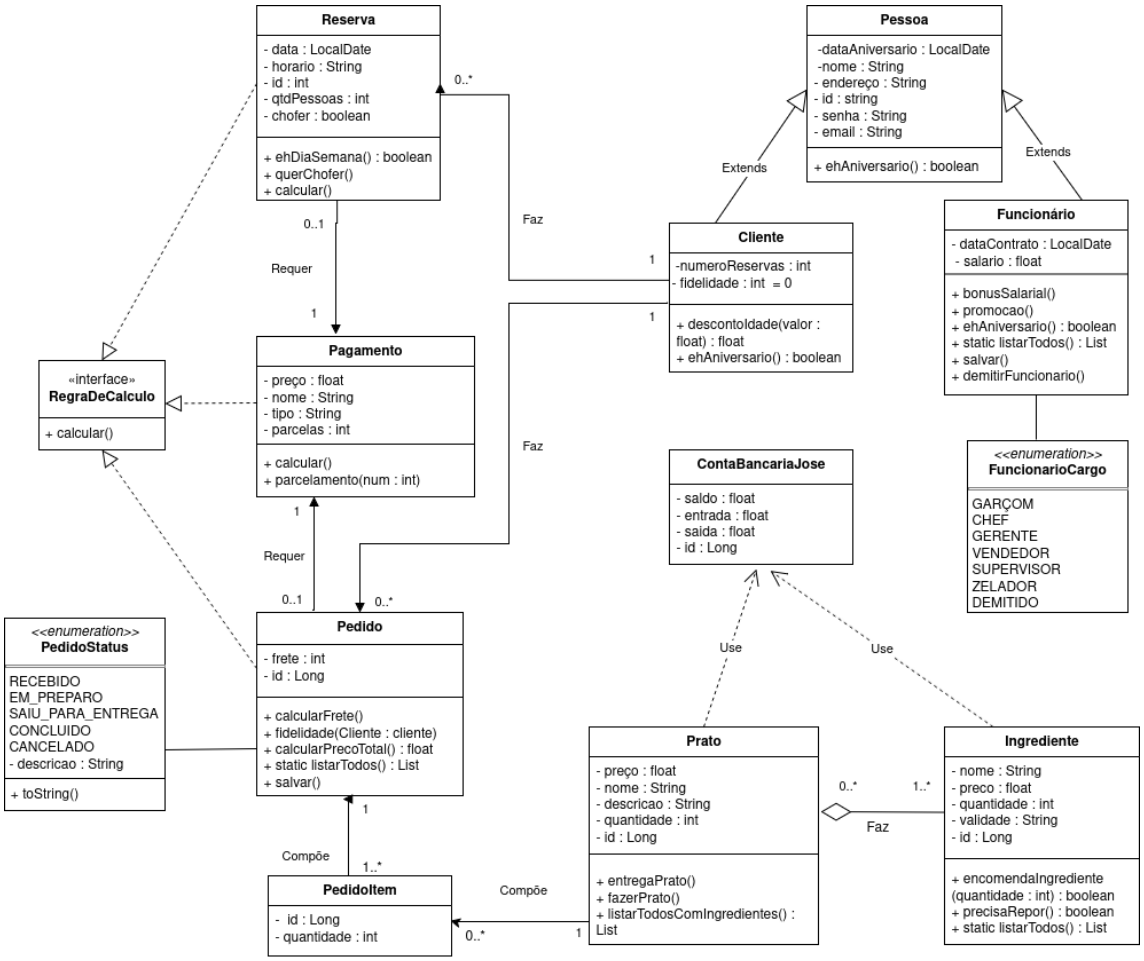
O restaurante Monsieur José adota regras de negócio que refletem a personalidade marcante e bem-humorada de seu fundador. Nas reservas, há incentivos para grupos e dias menos movimentados: clientes que reservam mesas durante a semana recebem descontos especiais, assim como grupos com mais de dez pessoas. Quando ambos os critérios são atendidos, os descontos são cumulativos, tornando a experiência ainda mais vantajosa.

Os clientes também contam com um programa de fidelidade. A cada pedido, pontos são acumulados em um cartão fidelidade, que podem ser trocados por descontos em pedidos ou reservas. José, apaixonado por celebrar a vida, oferece um desconto proporcional à idade do cliente — 1 real por ano de vida — e, no dia do aniversário, o cliente recebe um desconto especial ou até mesmo um prato gratuito. Para José, aniversários são datas sagradas.

No momento do pagamento, há estímulo ao uso de métodos práticos: quem paga via Pix recebe um desconto automático. Por outro lado, pagamentos parcelados sofrem a aplicação de juros altos, já que José tem uma notória aversão a compras a longo prazo.

Essas regras não apenas organizam o funcionamento do restaurante, mas traduzem sua essência: acolhedor, festivo e direto ao ponto.

3. Diagrama de Classes



3.1. Reserva

A classe **Reserva** é fundamental para a gestão de agendamentos no sistema, encapsulando todas as informações e dados pertinentes a uma reserva realizada por um cliente. Seus atributos incluem:

- **LocalDate data**: A data específica em que a reserva será efetuada.
- **String horário**: O horário agendado para a reserva.
- **boolean chofer**: Um indicador booleano que especifica se um chofer é necessário para a reserva.
- **int qtdPessoas**: A quantidade de pessoas envolvidas na reserva.

Além desses atributos diretos, a classe **Reserva** estabelece importantes relacionamentos com outras classes do sistema:

- **Cliente cliente**: Associa a reserva a um cliente específico, garantindo que a reserva seja vinculada ao seu responsável.
- **Pagamento pagamento**: Permite que informações de pagamento, como o valor final, sejam diretamente associadas e manipuladas no contexto da reserva.

A lógica de negócios da classe **Reserva** é implementada através de três métodos principais, que visam ajustar o preço da reserva de acordo com diferentes critérios:

- **ehDiaSemana()**: Aplica um desconto ao preço da reserva caso a data agendada caia em um dia de semana (de segunda a sexta-feira).
- **calcular()** (sobrescrito): Este método, que sobrescreve uma interface, recalcula o preço da reserva. Ele concede um desconto significativo se houver mais de cinco pessoas na reserva, refletindo uma política de preço para grupos maiores.
- **querChofer()**: Adiciona um valor fixo de R$10,00 ao preço da reserva se o serviço de chofer for solicitado pelo cliente.

Esses elementos permitem que a classe **Reserva** não apenas armazene dados, mas também atue dinamicamente na precificação, adaptando-se às necessidades e condições de cada agendamento.

3.2. Pessoa

A classe **Pessoa** é uma classe abstrata fundamental no sistema, servindo como base para a criação de outras entidades que representam indivíduos, como **Cliente** e **Funcionário**. Ela encapsula os atributos comuns a qualquer pessoa no sistema, garantindo a reutilização de código e uma estrutura de dados consistente. Seus atributos incluem:

- **String nome**: O nome completo da pessoa.
- **LocalDate dataAniversario**: A data de nascimento da pessoa.
- **String endereco**: O endereço completo da pessoa (opcional).
- **String senha**: A senha utilizada para autenticação no sistema.
- **String email**: O endereço de e-mail da pessoa, utilizado para login e comunicação.

Como uma classe abstrata, **Pessoa** não pode ser instanciada diretamente, mas fornece a estrutura e o comportamento básico que suas subclasses herdarão. Ela define um método abstrato que deve ser implementado por suas subclasses:

- **abstract boolean ehAniversario()**: Um método abstrato que deve ser implementado pelas subclasses para verificar se a data atual corresponde ao aniversário da pessoa.

A classe **Pessoa** também possui múltiplos construtores para facilitar a criação de objetos, permitindo a inicialização com ou sem o atributo de endereço, flexibilizando a criação de perfis.

3.3. Pagamento

A classe **Pagamento** é responsável por armazenar e gerenciar todas as informações relacionadas a um pagamento, bem como aplicar regras de cálculo para ajustar o preço final. Seus atributos incluem:

- **float preco**: O valor monetário do pagamento.
- **String nome**: O nome associado ao pagamento, que pode ser o nome do cliente ou uma descrição.
- **String tipo**: O método de pagamento utilizado (por exemplo, ”Pix”, ”Cartão de Crédito”).

- **int parcelas**: O número de parcelas em que o pagamento será dividido.

A classe **Pagamento** implementa a interface **RegraDeCalculo**, o que indica que ela possui lógica própria para alterar seu valor. Seus métodos principais são:

- **calcular()** (sobrescrito): Este método aplica um desconto de 10% ao **preco** se o **tipo** de pagamento for ”Pix”
- **parcelamento(int num)**: Aumenta o **preco** em 1% para cada parcela, simulando juros ou taxas de parcelamento.

A classe **Pagamento** é uma parte essencial do controle de custos e gastos, tanto para nossos clientes como para os gerentes.

3.4. Cliente

A classe **Cliente** estende a classe abstrata **Pessoa**, herdando seus atributos e comportamentos básicos. Ela representa os clientes do sistema, adicionando funcionalidades e dados específicos relacionados às interações de consumo. Seus atributos específicos incluem:

- **int numeroReservas**: A quantidade total de reservas realizadas pelo cliente.
- **int fidelidade**: Um indicador numérico que representa o nível de fidelidade do cliente ao longo do tempo.

Além dos atributos, a classe **Cliente** estabelece um relacionamento crucial com a classe **Reserva**:

- **List(Reservas) reservas**: Uma lista de reservas associadas a este cliente.

A classe **Cliente** também possui métodos importantes que adicionam lógica de negócio:

- **descontoIdade(float valor)**: Este método calcula um desconto no valor de um serviço com base na idade do cliente. Clientes com 70 anos ou mais recebem 20% de desconto, enquanto aqueles com 60 anos ou mais recebem 10% de desconto.
- **ehAniversario()** (sobrescrito): Sobrescreve o método abstrato da classe **Pessoa** para verificar se a data atual corresponde ao aniversário do cliente.

Através da herança e de seus atributos e métodos específicos, a classe **Cliente** modela de forma eficaz o papel dos usuários que realizam reservas e pedidos no sistema, permitindo o gerenciamento de sua fidelidade e a aplicação de políticas de desconto personalizadas.

3.5. Funcionário

A classe **Funcionário** estende a classe abstrata **Pessoa**, herdando as características básicas de um indivíduo e adicionando atributos e comportamentos específicos relacionados à sua função dentro da organização. Ela é crucial para o gerenciamento da equipe e suas respectivas responsabilidades e remunerações. Seus atributos específicos incluem:

- **FuncionarioCargo cargo**: O cargo atual do funcionário, representado por uma enumeração (**FuncionarioCargo**).
- **float salario**: O valor do salário do funcionário.

- **LocalDate dataContrato**: A data em que o funcionário foi contratado.

A classe **Funcionário** também incorpora métodos para interação com o banco de dados e regras de negócio:

- **listarTodos()**: Um método estático que busca e retorna uma lista de todos os funcionários cadastrados no banco de dados, independentemente do cargo (incluindo demitidos).
- **salvar()**: Persiste o estado atual do objeto **Funcionário** no banco de dados. Se o funcionário for novo (sem ID), ele é criado; caso contrário, é atualizado.
- **promocao()**: Implementa a lógica de promoção do funcionário, alterando seu **cargo** e aumentando seu **salario** com base em regras predefinidas (por exemplo, garçom para chef, vendedor para gerente).
- **demitirFuncionario()**: Altera o **cargo** do funcionário para **DEMITIDO** e zera seu **salario**. Esta alteração requer uma chamada subsequente ao método **salvar()** para ser persistida no banco de dados.
- **ehAniversario()** (sobrescrito): Sobrescreve o método abstrato da classe **Pessoa** para verificar se a data atual corresponde ao aniversário do funcionário.

A classe Funcionário é essencial para a gerência conseguir manter o controle correto acerca dos contratados e demitidos da empresa.

3.6. Pedido

A classe **Pedido** representa um pedido realizado por um cliente no sistema. Ela consolida informações sobre os itens solicitados, o pagamento, o cliente associado e o status do pedido, além de gerenciar a lógica de cálculo de valores e frete. Seus atributos incluem:

- **Pagamento pagamento**: Um objeto de relação com a classe **Pagamento** que contém todos os detalhes do pagamento associado a este pedido.
- **List(PedidoItem) itensPedido**: Uma lista de itens da classe **PedidoItem** que vai relacionar cada pedido a uma quantidade de pratos que o compõem.
- **Cliente consumidor**: O cliente que realizou o pedido.
- **int frete**: O valor do frete associado à entrega do pedido.
- **PedidoStatus status**: O status atual do pedido, representado por uma enumeração (**PedidoStatus**), que indica a fase em que o pedido se encontra (e.g., RECEBIDO, EM PREPARACAO, ENTREGUE).

A classe **Pedido** também provê métodos para interação com o banco de dados e regras de negócio:

- **calcularFrete()**: Calcula o valor do frete com base no endereço do cliente (**consumidor**). Diferentes zonas geográficas (Centro, Zona Leste, Zona Sul, Zona Norte, Zona Oeste) têm valores de frete distintos.
- **calcularPrecoTotal()**: Calcula o preço total do pedido, somando o preço de cada item (quantidade * preço do prato) e adicionando o valor do frete.
- **fidelidade(Cliente cliente)**: Aplica um desconto ao preço total do pedido com base na fidelidade do cliente. Para cada ponto de fidelidade do cliente, é aplicado um desconto de 5% sobre o preço acumulado.

A classe **Pedido** é central para o fluxo de operações do sistema, coordenando os itens, o pagamento e o cliente para um processamento eficiente e completo dos pedidos.

3.7. ContaBancariaJose

A classe **ContaBancariaJose** representa a conta bancária principal do restaurante, designada como a conta do ”Sr. José”, onde os recursos financeiros são gerenciados. Esta classe é crucial para o controle de fluxo de caixa, registrando todas as movimentações financeiras. Seus atributos incluem:

- **float saldo**: O saldo atual da conta bancária do restaurante.
- **float entrada**: O valor total das entradas de dinheiro na conta.
- **float saida**: O valor total das saídas de dinheiro da conta.

A classe **ContaBancariaJose** é uma entidade persistente (**@Entity**), o que significa que seu estado pode ser armazenado e recuperado de um banco de dados. Ela fornece os métodos básicos para acessar e modificar o saldo, as entradas e as saídas, permitindo o registro e a consulta das transações financeiras do restaurante.

Embora o código fornecido não inclua métodos explícitos para realizar depósitos ou saques que alterem o saldo, a presença dos atributos **entrada** e **saida** sugere que o controle financeiro é feito através do registro desses movimentos, permitindo que o **saldo** seja atualizado de acordo com a lógica de negócio implementada em outras partes do sistema.

3.8. Prato

A classe **Prato** representa um item de menu disponível no restaurante. Ela armazena informações detalhadas sobre cada prato, incluindo seu preço, composição e disponibilidade em estoque. Seus atributos incluem:

- **float preco**: O preço de venda do prato.
- **String nome**: O nome do prato, que deve ser único.
- **String descricao**: Uma descrição detalhada do prato (opcional).
- **int quantidade**: A quantidade de pratos prontos disponíveis em estoque.

A classe **Prato** também estabelece um relacionamento crucial com a classe **Ingrediente**:

- **List(Ingrediente) ingredientes**: Uma lista dos ingredientes que compõem o prato. O relacionamento indica que vários pratos podem usar os mesmos ingredientes e vice-versa, e que a persistência de um prato também pode persistir novos ingredientes. A tabela de junção **prato_ingrediente** gerencia essa associação.

A classe **Prato** oferece métodos para interação com o banco de dados e lógica de negócio essencial para o funcionamento do restaurante:

- **fazPrato()**: Este método simula a ação de ”cozinhar”um prato. Ele verifica a disponibilidade dos **ingredientes** necessários e, se houver estoque suficiente, decrementa a quantidade de cada ingrediente utilizado e incrementa a **quantidade** de pratos prontos em estoque. Todas as alterações são persistidas no banco de dados dentro de uma transação.
- **entregaPrato()**: Simula a ”entrega”ou venda de um prato. Primeiramente, verifica se há estoque disponível do prato. Se houver, decrementa a **quantidade** em estoque do prato e registra o **preco** do prato como uma **entrada** na **ContaBancariaJose** do restaurante. A transação garante que ambas as operações (diminuição de estoque e registro financeiro) sejam realizadas atomicamente.

A classe **Prato** é central para o controle do inventário de pratos prontos e para as operações de produção e venda no restaurante, integrando a gestão de ingredientes e o fluxo financeiro.

3.9. Ingrediente

A classe **Ingrediente** representa um item utilizado na preparação dos pratos do restaurante. Ela é essencial para o gerenciamento do estoque, controle de custos e validade dos produtos. Seus atributos incluem:

- **String nome**: O nome do ingrediente (ex: "Tomate", "Farinha").
- **float preco**: O preço unitário de compra do ingrediente.
- **int quantidade**: A quantidade disponível em estoque do ingrediente.
- **LocalDate validade**: A data de validade do ingrediente.

A classe **Ingrediente** é uma entidade persistente (**@Entity**), o que permite que suas informações sejam armazenadas e recuperadas de um banco de dados. Ela oferece métodos para consulta e manipulação de seu estado, além de funcionalidades para o controle de estoque e reabastecimento:

- **listarTodos()**: Um método estático que busca e retorna uma lista de todos os ingredientes registrados no banco de dados.
- **precisaRepor()**: Um método que verifica se a quantidade atual do ingrediente em estoque está abaixo de um limite (neste caso, 10 unidades), indicando a necessidade de reposição.
- **encomendaIngrediente(int quantidadeAComprar)**: Este método processa a compra de uma determinada quantidade do ingrediente. Ele verifica se há saldo suficiente na **ContaBancariaJose** para cobrir o custo da encomenda. Se houver, o valor correspondente é deduzido do saldo da conta, a saída é registrada, e a **quantidade** do ingrediente em estoque é atualizada. Todas essas operações são realizadas dentro de uma transação para garantir a consistência dos dados no banco.

A classe **Ingrediente** é vital para a operação diária do restaurante, garantindo que os ingredientes necessários para a produção dos pratos estejam sempre disponíveis e que o controle financeiro das compras seja rigoroso.

3.10. Enums

As enumerações (ou **enums**) são tipos especiais de classes que representam um conjunto fixo de constantes. No sistema, elas são utilizadas para padronizar e restringir os valores de determinados atributos, tornando o código mais legível, seguro e fácil de manter.

Enum FuncionarioCargo

A enumeração **FuncionarioCargo** define os possíveis cargos que um funcionário pode ter dentro do restaurante. Ela garante que os cargos sejam sempre um dos valores predefinidos, evitando erros de digitação e inconsistências. Os cargos definidos são:

- **GERENTE**: Responsável pela gestão geral e operações.

- **VENDEDOR**: Encarregado das vendas e atendimento ao cliente.
- **SUPERVISOR**: Supervisiona equipes e processos específicos.
- **GARCOM**: Atende clientes e serve pratos.
- **CHEF**: Responsável pela cozinha e criação de pratos.
- **ZELADOR**: Cuida da limpeza e manutenção do ambiente.
- **DEMITIDO**: Indica que o funcionário não faz mais parte da equipe ativa.

Enum PedidoStatus

A enumeração **PedidoStatus** representa os diferentes estágios pelos quais um pedido pode passar desde a sua criação até a conclusão ou cancelamento. Cada status possui uma descrição mais amigável para exibição. Os status definidos são:

- **RECEBIDO** (”Recebido”): O pedido foi efetuado e registrado no sistema.
- **EM_PREPARO** (”Em Preparo”): O pedido está sendo preparado na cozinha.
- **SAIU_PARA_ENTREGA** (”Saiu para Entrega”): O pedido está a caminho do cliente.
- **CONCLUIDO** (”Concluído”): O pedido foi entregue com sucesso.
- **CANCELADO** (”Cancelado”): O pedido foi cancelado por algum motivo.

A utilização dessas enumerações contribui para a clareza do código e para a integridade dos dados, ao limitar as opções de valores para atributos cruciais do sistema.

3.11. Interface RegraDeCalculo

A interface **RegraDeCalculo** define um contrato para classes que necessitam implementar uma lógica de cálculo específica. Interfaces são essenciais em programação orientada a objetos para promover o polimorfismo e garantir que certas classes forneçam uma implementação para um método comum.

A interface **RegraDeCalculo** possui um único método:

- **void calcular()**: Este método abstrato deve ser implementado por qualquer classe que deseje aderir a esta interface. Ele indica que a classe em questão possui uma funcionalidade de cálculo que será executada. No contexto deste sistema, as classes **Pagamento** e **Reserva** implementam esta interface para aplicar suas próprias lógicas de ajuste de valores.

A utilização desta interface permite que o sistema trate diferentes tipos de objetos (como Pagamentos e Reservas) de forma uniforme quando a operação de cálculo for necessária, sem precisar conhecer os detalhes internos de como cada um realiza seu cálculo.

3.12. PedidoItem

A classe **PedidoItem** representa um item individual dentro de um **Pedido**. Ela atua como uma entidade de junção entre um pedido e os pratos que o compõem, armazenando a quantidade de cada prato solicitado em um pedido específico. Seus atributos incluem:

- **Long id**: Um identificador único para o item do pedido, gerado automaticamente.
- **int quantidade**: A quantidade do **Prato** específico que foi solicitada neste item do pedido.

A classe **PedidoItem** estabelece importantes relacionamentos com outras entidades:

- **Pedido pedido**: Associa este item a um **Pedido** específico. O relacionamento **@ManyToOne** com **fetch = FetchType.LAZY** indica que muitos itens podem pertencer a um único pedido, e o pedido é carregado sob demanda. A coluna **pedido_id** atua como chave estrangeira.
- **Prato prato**: Associa este item a um **Prato** específico do cardápio. O relacionamento **@ManyToOne** com **fetch = FetchType.EAGER** indica que muitos itens podem se referir ao mesmo prato, e o prato será carregado imediatamente junto com o item do pedido. A coluna **prato_id** atua como chave estrangeira.

A classe **PedidoItem** é uma entidade persistente (**@Entity**) e é utilizada para criar uma tabela auxiliar (**pedido_itens**) no banco de dados, que gerencia a relação entre pedidos e pratos, permitindo que um pedido contenha múltiplos pratos e que a quantidade de cada prato seja especificada.

3.13. Mapeamento Objeto-Relacional (ORM)

O banco de dados utilizado, **restauranteMonsieur**, foi estruturado com base em um mapeamento Objeto-Relacional (ORM). Isso significa que cada classe Java do sistema que representa uma entidade de negócio é automaticamente mapeada para uma tabela correspondente no banco de dados.

As classes que foram detalhadas anteriormente (como **Cliente**, **Funcionario**, **Ingrediente**, **Pedido**, **Prato**, **Reserva** e **ContaBancariaJose**), bem como as tabelas de junção necessárias para relacionamentos *Many-to-Many* (como **prato_ingrediente** e **pedido_itens**), são criadas e gerenciadas de forma transparente pelo framework ORM (como o JPA/Hibernate).

Essa abordagem simplifica drasticamente o desenvolvimento, pois os desenvolvedores podem trabalhar com objetos em Java, e o ORM se encarrega de persistir esses objetos no banco de dados relacional e de recuperá-los quando necessário, convertendo-os de volta para objetos. Isso elimina a necessidade de escrever grande parte do código SQL manual para operações de CRUD (Create, Read, Update, Delete), focando mais na lógica de negócio da aplicação.

As tabelas visíveis no banco de dados, como **cliente**, **contabancaria**, **funcionario**, **ingredientes**, **pedido_itens**, **pedidos**, **pessoas**, **prato_ingrediente**(criada apenas no banco, não possui classe própria), **pratos** e **reservas**, são o resultado direto deste mapeamento ORM, refletindo a estrutura das classes Java e seus relacionamentos.

4. Telas

Esta seção apresenta as principais telas do sistema, detalhando suas funcionalidades e o fluxo de interação do usuário. A seguir, cada interface será demonstrada visualmente e descrita.

4.1. Tela Inicial

A Figura 1 exibe a tela inicial do sistema, que serve como portal de entrada para as principais funcionalidades. O design é minimalista e focado na usabilidade, com um fundo em tons de vinho que remete à identidade visual do restaurante.

Ao centro, destacam-se dois grandes painéis de acesso:

- **Reservas:** Leva o usuário a uma área onde é possível realizar as reservas de mesas.
- **Delivery:** Direciona o usuário ao cardápio digital, onde ele pode escolher sua refeição no conforto de sua residência.

No topo da tela, o ícone de perfil dá acesso à área do gerente e suas funcionalidades administrativas, enquanto o ícone de bandeira possibilita a alternância de idioma.

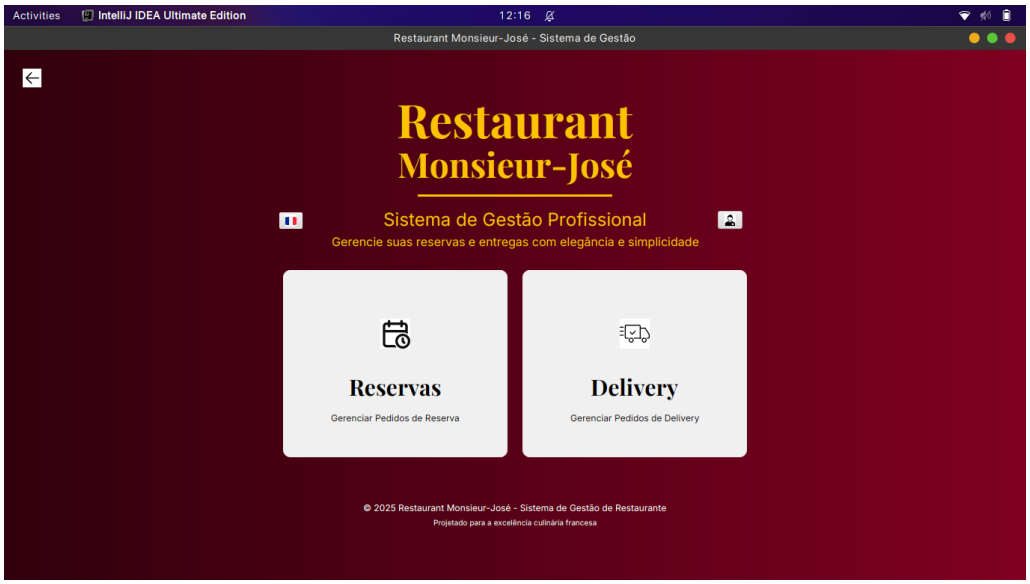


Figura 1. Tela inicial do sistema.

4.2. Tela de Login e Criação de Conta

A Figura 2 apresenta a interface de autenticação e cadastro, projetada para os clientes do restaurante. A tela é dividida em duas seções verticais, mantendo a identidade visual do sistema e facilitando a navegação do usuário.

- **Criar Conta:** Esta seção é destinada aos novos clientes. O formulário solicita dados essenciais para o cadastro, como nome completo, data de aniversário, endereço, e-mail e senha. A solicitação da data de nascimento é um ponto funcional importante, pois se conecta diretamente à regra de negócio que concede descontos a aniversariantes.

- **Entrar:** Clientes já cadastrados podem acessar suas contas utilizando e-mail e senha. Para maior comodidade, a interface oferece a opção ”Lembrar de mim”e um link para recuperação de senha (”Esqueceu a senha?”), garantindo que o acesso seja simples e seguro.

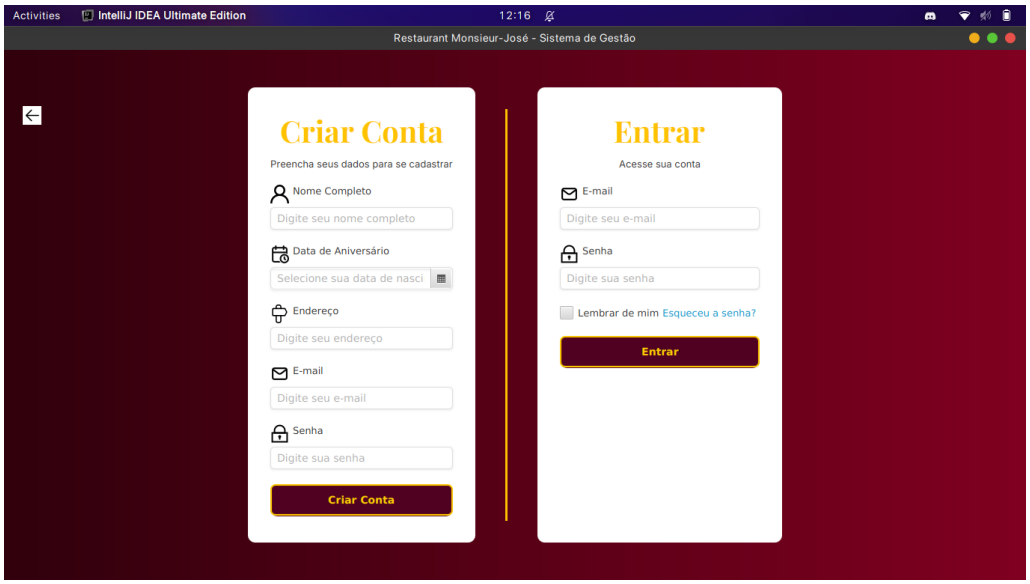


Figura 2. Interface para login e criação de conta de cliente.

4.3. Tela de Reserva de Mesa

Esta tela, ilustrada na Figura 3, é dedicada ao processo de reserva de mesas, um dos fluxos centrais para o cliente. A interface foi projetada para ser direta e informativa, guiando o usuário no preenchimento dos dados necessários para garantir sua mesa no restaurante.

O formulário ”Informações da Reserva”solicita os seguintes dados:

- **Nome Completo e Email:** Para identificação e contato.
- **Horário e Data:** Permitem ao cliente selecionar o momento exato de sua visita.
- **Número de Pessoas:** Essencial para o planejamento e alocação de mesas pelo restaurante.
- **Tipo de Pagamento:** Um campo de seleção que já informa sobre as vantagens de cada método, como o desconto de 10% para pagamentos via Pix, aplicando diretamente uma das regras de negócio do sistema.
- **Chofer:** Uma checkbox que oferece um serviço adicional e exclusivo de chofer, agregando valor à experiência do cliente.

Ao final, o botão ”Confirmar reserva”submete as informações, finalizando o processo de agendamento de forma rápida e eficiente.

Figura 3. Interface para a realização de reservas de mesa.

4.4. Tela de Delivery

A Figura 4 ilustra a interface de pedidos para delivery, que combina o cardápio e o carrinho de compras em uma única tela para uma experiência de usuário fluida e integrada. O layout é dividido em duas seções principais.

- **Cardápio:** Esta seção apresenta uma lista rolável com todos os itens disponíveis para entrega. Cada item é exibido com seu nome, uma breve descrição, o preço e um campo para selecionar a quantidade desejada. Com um clique no botão ”Adicionar”, o cliente insere o item em seu pedido.

- **Seu Pedido:** Funcionando como um carrinho de compras, esta área exibe um resumo dinâmico do pedido. Os itens adicionados são listados com sua quantidade e subtotal. O cliente pode ajustar a quantidade ou remover itens diretamente desta seção. Ao final, são exibidos o valor total, o campo para seleção da modalidade de pagamento — que novamente aplica a regra de negócio do desconto para Pix — e o botão ”Finalizar Pedido”para concluir a compra.

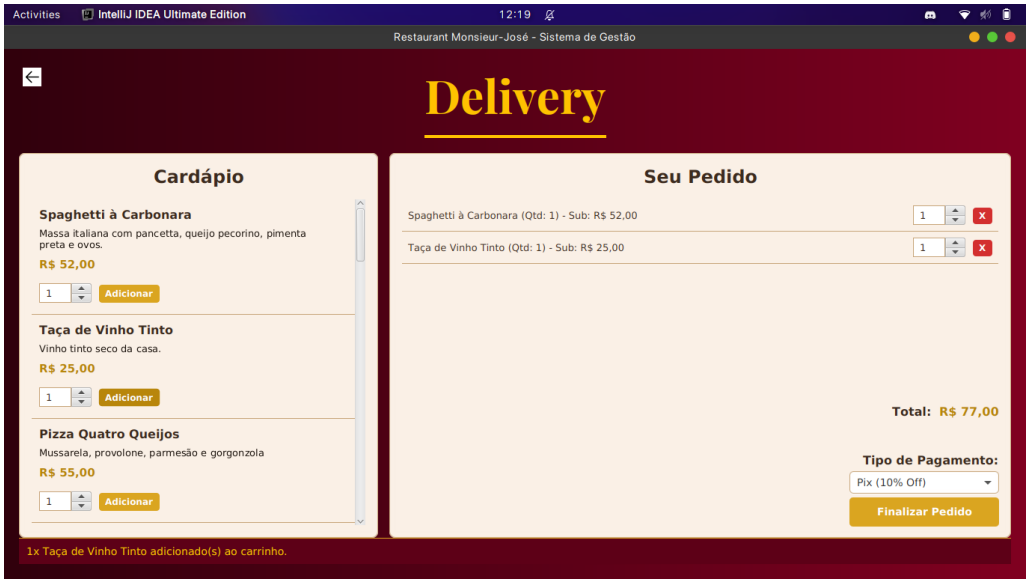


Figura 4. Interface de seleção de itens e finalização de pedido para delivery.

4.5. Tela de Finalização de Pagamento

A Figura 5 demonstra a tela de finalização de pagamento, a última etapa do processo de pedido ou reserva. A interface foi estruturada para ser transparente e segura, consolidando todas as informações para que o cliente possa revisar e confirmar sua transação com confiança.

- **Detalhes do Pedido:** Esta seção é designada para exibir um resumo detalhado dos itens do pedido de delivery ou das especificidades da reserva.
- **Forma de Pagamento:** Aqui, o cliente interage com os aspectos financeiros da transação. O campo ”Resumo do Pagamento”é o elemento central, detalhando subtotais e o **Valor Total**, no qual o sistema aplica automaticamente todos os descontos aplicáveis.

Figura 5. Interface de finalização de pagamento com descontos aplicados.

4.6. Painel Administrativo de Serviços

Acessada pelo ícone de perfil, com a senha PSG5-0 a seção de Serviços (Figura 6) funciona como o painel de controle para o gerente. Esta interface modular oferece acesso rápido a todas as áreas vitais da administração.

- **Cadastros:** Para o gerenciamento de contas de clientes e funcionários.
- **Pedidos:** Centraliza o monitoramento de todos os pedidos de delivery.
- **Reservas:** Permite visualizar e administrar todas as reservas.
- **Estoque:** Ferramenta para o controle de suprimentos e ingredientes.
- **Conta:** Módulo dedicado à gestão financeira.
- **Cardápio:** Permite ao gerente visualizar o menu.

Figura 6. Painel de controle administrativo.

4.7. Tela de Visualização do Cardápio

A Figura 7 exibe a interface de visualização do cardápio. Cada item é apresentado em um card individual, contendo nome, descrição e preço. De forma criativa, os pratos recebem nomes de jogadores de futebol franceses, reforçando o tema do restaurante.

Figura 7. Interface de visualização do cardápio do restaurante.

4.8. Tela de Análise Financeira

A tela da Figura 8 é uma ferramenta de inteligência de negócios para a gestão. Acessada pelo painel administrativo, ela oferece uma visão da saúde financeira do estabelecimento. O gráfico de barras compara os ganhos (barras verdes) e as despesas (barras vermelhas) de cada mês.

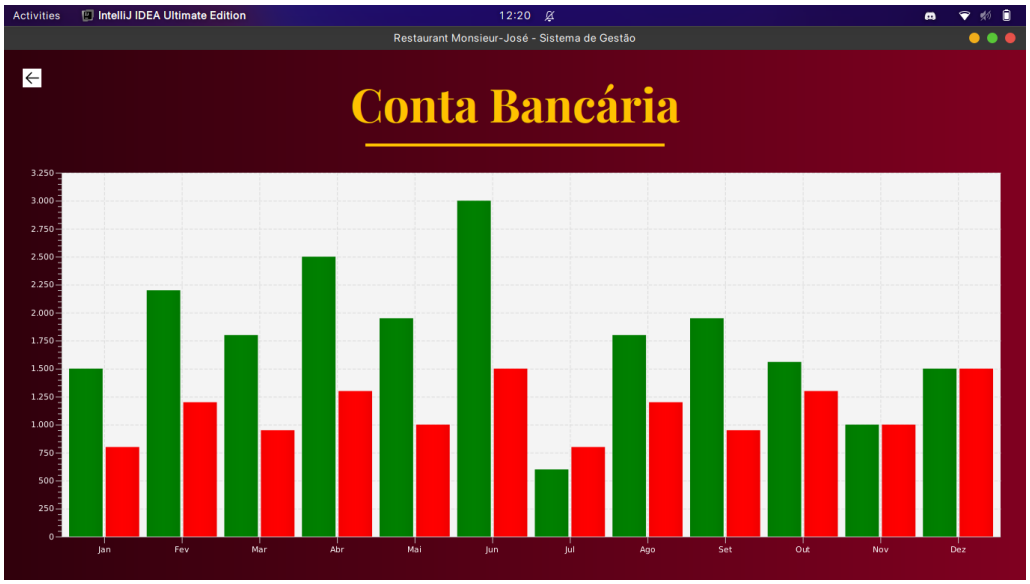


Figura 8. Gráfico de análise financeira com ganhos e despesas mensais.

4.9. Tela de Gestão de Estoque e Cozinha

A Figura 9 apresenta a tela de ”Estoque e Cozinha”. A seção ”Ingredientes”detalha os insumos, quantidade e um botão ”Pedir”para reabastecimento. A seção ”Pratos”informa

quantos de cada item estão preparados, com um botão ”Cozinhar”que atualiza o estoque de ingredientes.

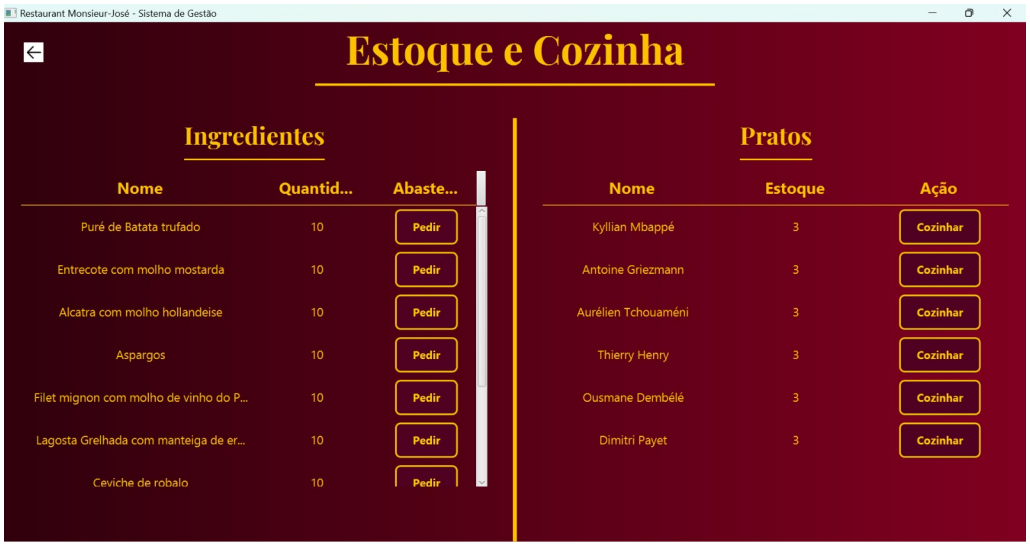


Figura 9. Interface de controle de estoque e produção de pratos.

4.10. Tela de Gerenciamento de Reservas

A Figura 10 ilustra a tela de gerenciamento de reservas. A interface apresenta uma lista de agendamentos com cliente, data, horário, nº de pessoas e se o serviço de chofer foi solicitado. Filtros por nome e data otimizam a consulta.

Figura 10. Interface para a consulta e gerenciamento de reservas.

4.11. Tela de Gerenciamento de Pedidos

A Figura 11 apresenta o painel de controle para ordens de delivery. A tela é dividida em uma lista de pedidos (com cliente, valor e status) e um painel de detalhes, onde o gerente pode ver os itens e alterar o status do pedido (ex: ”Recebido”, ”Em Preparo”).

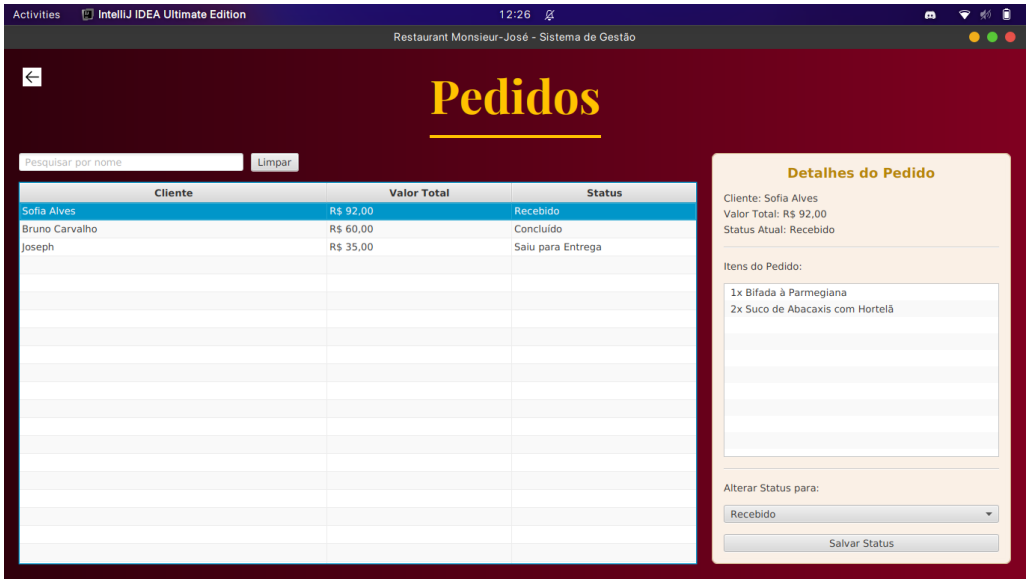


Figura 11. Interface para o gerenciamento e acompanhamento de pedidos.

4.12. Tela de Gestão de Contas e Cadastros

A Figura 12 serve como portal para o gerenciamento de usuários, refletindo a estrutura de classes onde ”Clientes”e ”Funcionários”são especializações de ”Pessoa”.

Figura 12. Interface de acesso aos cadastros de Clientes e Funcionários.

4.13. Tela de Gerenciamento de Clientes

A Figura 13 apresenta um panorama da base de clientes, com nome, email, aniversário (para aplicar descontos) e endereço (para delivery). Um filtro por nome agiliza a busca.

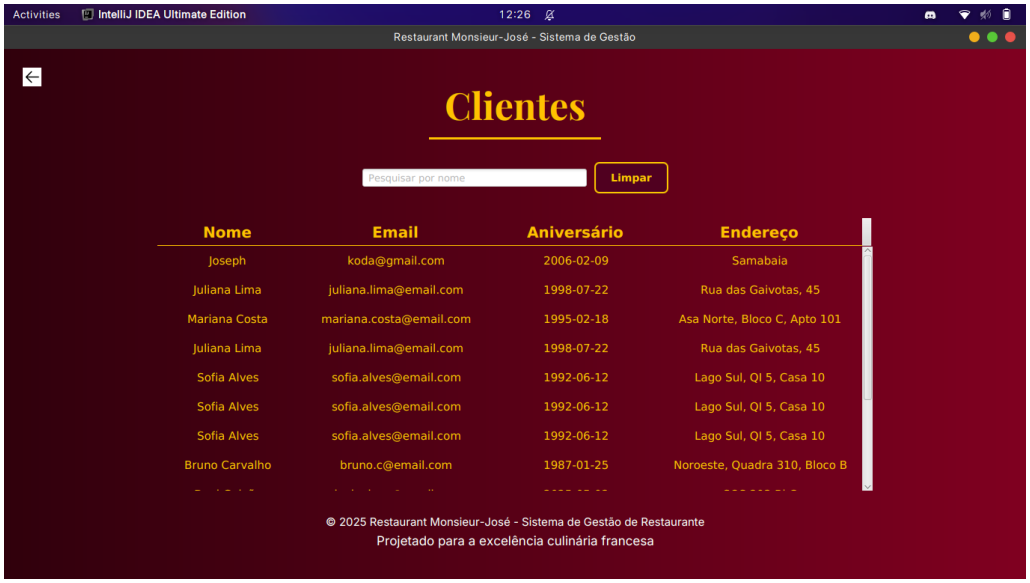


Figura 13. Interface de consulta e gerenciamento do cadastro de clientes.

4.14. Tela de Gerenciamento de Funcionários

A Figura 14 detalha a tela de RH. Um painel de gestão lista os funcionários com seus cargos e permite ações de ”Promover”e ”Demitir”. Outro painel, de ”Contratar”, oferece um formulário para adicionar novos colaboradores ao sistema.

Figura 14. Interface para o gerenciamento do ciclo de vida dos funcionários.

5. Bibliografia

1. WEISFELD, Matt. *The Object-Oriented Thought Process*. 4ª ed. Addison-Wesley Professional, 2013.
2. MEYER, Bertrand. *Object-Oriented Software Construction*. 2ª ed. Prentice Hall.
3. BOOCH, Grady et al. *Object-Oriented Analysis and Design with Applications*. Addison-Wesley Professional, 2007.