Centro Universitário de Excelência
Sistemas de Informação

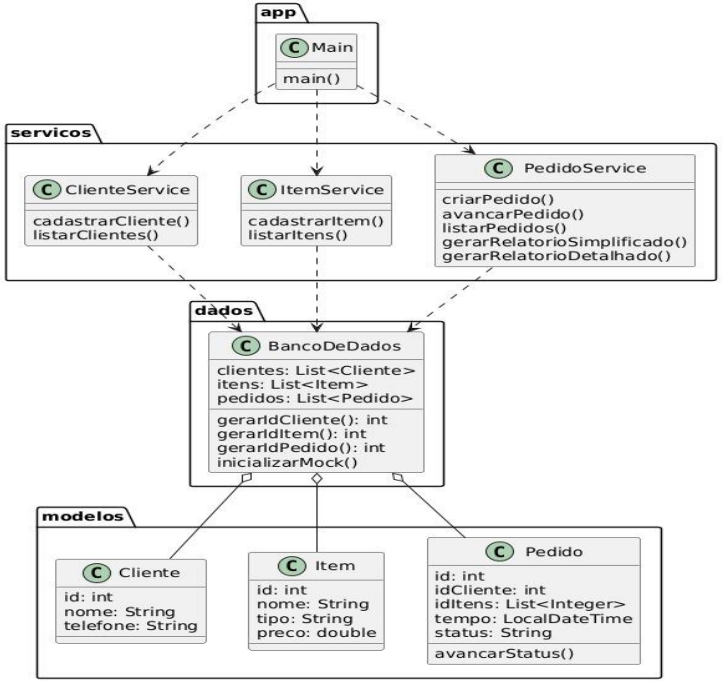
Desenvolvimento web

Orientada A Objetos

Autores: <Samira Santos De Jesus>
<Erick Dos Santos Rezende>
<Victor Rogerio Aguiar Do Rosário>
<Laysa Lima De Pinho>
<Vinicius De Oliveira Costa>

Feira de Santana, 2025

Diagrama de Classes



Entidades:
`Cliente(id, nome, telefone)`
`Item(id, nome, tipo, preco)`
`Pedido(id, idCliente, idItens, tempo, status)`

Regras-chave do Pedido: nasce aceito (na confirmação), registra data/hora, e avança por
ACEITO → PREPARANDO → FEITO → AGUARDANDO ENTREGADOR → SAIU PARA ENTREGA → ENTREGUE.

Serviços (casos de uso):
`ClienteService`/`ItemService` (cadastro/lista) e `PedidoService` (criar pedido, avançar status, listar por status, relatórios simplificado/detalhado).

Menu

```
===== MENU =====
1 - Listar Clientes
2 - Cadastrar Cliente
3 - Listar Itens
4 - Cadastrar Item
5 - Criar Pedido
6 - Avançar Status do Pedido
7 - Listar Pedidos por Status
8 - Gerar relatorio simplificado
9 - Gerar relatório Detalhado
0 - Sair
Escolha:
```

Listar Clientes – mostra todos os clientes cadastrados

Cadastrar Cliente – registra um novo cliente
Listar Itens – exibe o cardápio .
Cadastrar Item – inclui um novo item no cardápio
Criar Pedido – seleciona o cliente, adiciona itens do cardápio
Avançar Status do Pedido – localiza o pedido pelo número e aplica a **próxima etapa** do fluxo
Listar Pedidos por Status – filtra e mostra os pedidos em um status informado (ex.: “PREPARANDO”).
Gerar relatório simplificado – calcula a quantidade de pedidos do dia e faturamento total.
Gerar relatório detalhado – lista os pedidos do dia com cliente, itens e total por pedido.
Sair – encerra a aplicação.

Cadastro e lista de Clientes

```
--- CLIENTES ---
ID: 1 | Nome: Erick Rezende | Telefone: 99999-1111
ID: 2 | Nome: Laysa Lima | Telefone: 98888-2222
ID: 3 | Nome: Vinicios Costa | Telefone: 97777-3333
```

```
Escolha: 2
Digite o nome do cliente: Samira
Digite o telefone do cliente: 75981908860
Cliente cadastrado com sucesso! ID: 4
```

Cadastro (Menu 2 – Cadastrar Cliente)
CLI pede **nome** e **telefone**.
Faz validações básicas de entrada.
`BancoDeDados.gerarIdCliente()` cria um **ID sequencial e único**.
Cria o objeto `Cliente(id, nome, telefone)` e adiciona a lista.
Exibe confirmação: *"Cliente cadastrado com sucesso! ID: X"*.
Observação: os dados ficam **em memória** durante a execução.
Listagem (Menu 1 – Listar Clientes)
Percorre `BancoDeDados.clientes` e imprime cada registro no formato:
`ID: <id> | Nome: <nome> | Telefone: <telefone>`
(se a lista estiver vazia, informa que não há clientes cadastrados).

Cadastro e lista de itens

```
Escolha: 4
Digite o nome do item: Frango assado
Digite o tipo do item: Comida
Digite o preço do item: 22,90
Item cadastrado com sucesso! ID: 4
```

```
--- ITENS ---
ID: 1 | Nome: Bolo de chocolate | Tipo: Sobremesa | Preço: R$ 35.9
ID: 2 | Nome: Hambúrguer | Tipo: Comida | Preço: R$ 22.5
ID: 3 | Nome: Coca-Cola 2L | Tipo: Bebida | Preço: R$ 10.0
ID: 4 | Nome: Frango assado | Tipo: Comida | Preço: R$ 22.9
```

Cadastro (Menu 4 – Cadastrar Item)
CLI pede **nome**, **tipo** e **preço**.
Faz **validações básicas** (nome não vazio; preço numérico/positivo).
`BancoDeDados.gerarIdItem()` cria um **ID sequencial e único**.
Cria o objeto `Item(id, nome, tipo, preco)` e **adiciona à lista**.
Exibe confirmação: **"Item cadastrado com sucesso! ID: X"**.
Observação: os dados ficam **em memória**; preço é armazenado como `double`.
Listagem (Menu 3 – Listar Itens)
Percorre `BancoDeDados.itens` e imprime cada registro no formato:
`ID: <id> | Nome: <nome> | Tipo: <tipo> | Preço: R$ <preco>`
(Se a lista estiver vazia, informa que **não há itens cadastrados**.)

Criar Pedido

```
Escolha: 5

--- CLIENTES ---
ID: 1 | Nome: Erick Rezende | Telefone: 99999-1111
ID: 2 | Nome: Laysa Lima | Telefone: 98888-2222
ID: 3 | Nome: Vinicios Costa | Telefone: 97777-3333
ID: 4 | Nome: Samira | Telefone: 75981908860

Digite o ID do cliente: 4

--- ITENS ---
ID: 1 | Nome: Bolo de chocolate | Tipo: Sobremesa | Preço: R$ 35.9
ID: 2 | Nome: Hambúrguer | Tipo: Comida | Preço: R$ 22.5
ID: 3 | Nome: Coca-Cola 2L | Tipo: Bebida | Preço: R$ 10.0
ID: 4 | Nome: Frango assado | Tipo: Comida | Preço: R$ 22.9

Digite o ID do item: 1
Quantidade: 1
(1) Adicionar mais itens:
(0) Não adicionar mais itens:
Escolha: 0
Pedido criado com sucesso!
```

Criar Pedido (Menu 5 – Criar Pedido)
CLI **lista os clientes** e pede o **ID do cliente**.
Mostra o **cardápio** e pede o **ID do item** para adicionar
Digite **1** para adicionar um novo item pelo respectivo ID, **0** para confirmar pedido
Cada inclusão é gravada em uma lista `idItens` (ex.: `[3, 3, 7]`).
`BancoDeDados.gerarIdPedido()` cria um **nº sequencial**.
Cria `Pedido(id, idCliente, idItens, tempo = LocalDateTime.now(), status = "ACEITO")`.
Adiciona em `BancoDeDados.pedidos` e mostra: **‘’Pedido criado com sucesso!”**

Avançar status

```
Escolha: 6
Digite o ID do pedido: 1
Novo status: Preparando
```

```
Escolha: 6
Digite o ID do pedido: 1
Novo status: Feito
```

```
Escolha: 6
Digite o ID do pedido: 1
Novo status: Aguardando entregador
```

```
Escolha: 6
Digite o ID do pedido: 1
Novo status: Saiu para entrega
```

```
Escolha: 6
Digite o ID do pedido: 1
Novo status: Entregue
```

```
Escolha: 6
Digite o ID do pedido: 1
Pedido já finalizado.
Novo status: Entregue
```

Avançar Status do Pedido (Menu 6 – Avançar Status)
CLI solicita o **ID do pedido** e **valida** se existe em `BancoDeDados.pedidos`.
Localiza o objeto `Pedido` e verifica o **status atual**.
Se já estiver **ENTREGUE**, informa que **"Pedido já finalizado"**
Caso contrário, chama `Pedido.avancarStatus()`
ACEITO → PREPARANDO → FEITO → AGUARDANDO ENTREGADOR → SAIU PARA ENTREGA → ENTREGUE.
(Não permite pular etapas nem retroceder.)
O pedido é **atualizado em memória**
Exibe confirmação: **"Novo status: "X"**.

Listar pedidos por status

```
Escolha: 7
Escolha o status buscado:
1 - Aceito
2 - Preparando
3 - Feito
4 - Aguardando entregador
5 - Saiu para entrega
6 - Entregue
Escolha: 6

--- PEDIDOS ---
Pedido ID: 1 | Cliente ID: 4 | Item ID: [1] | Data e hora: 30-08-2025 21:35:38 | Status: Entregue
```

Listar pedidos por status (Menu 7 – Listar Pedidos por Status)
Exibe um **menu de status** numerado (1 a 6).
Lê a opção com `Scanner` e, via `switch`, mapeia para a **string** do status buscado
Percorre `BancoDeDados.pedidos`; para cada `Pedido` cujo `pedido.status` **é igual** ao `statusBuscado` e imprime: `Pedido ID`, `Cliente ID`, `Item ID(s)` (lista), **data/hora** formatada com `DateTimeFormatter("dd-MM-yyyy HH:mm:ss")` e o **status**.
Usa a flag `isEncontrado` para saber se algum pedido foi exibido; caso **nenhum** corresponda, mostra:
"Nenhum pedido encontrado com o status <status>."
Tratamento de erros: bloco `try/catch` captura entradas não numéricas e exibe
"Erro: entrada inválida. Por favor, digite um número."

Relatório simplificado

```
Escolha: 8

=== Relatório Simples 30/08/2025 ===
Total de pedidos: 2
Valor total dos pedidos: 35,90
```

Gerar Relatório Simplificado (Menu 8)
Data do cabeçalho: pega `LocalDate.now()`, formata em `"dd/MM/yyyy"` e imprime o título
Total de pedidos: `totalPedidos = BancoDeDados.pedidos.size()` (conta quantos `Pedido` existem).
Somatório de valores: inicializa `valorTotal = 0.0` e:
percorre **cada `Pedido`**;
Para cada **id** em `pedido.idItens` (uma ocorrência **= 1 unidade** do item),
Busca o **Item** correspondente em `BancoDeDados.itens` (comparando `item.id == idItem`) e soma `item.preco` a `valorTotal`.
Saída: imprime
`Total de pedidos: <n>` e
`Valor total dos pedidos: <valor>` (formatado com 2 casas)

Relatório Detalhado

```
=== Relatório Detalhado ===

Pedido ID: 1
Data: 30/08/2025 22:01
Cliente ID: 3 | Nome: Vinicios Costa | Telefone: 97777-3333
Itens:
Coca-Cola 2L 5x | R$ 50,00
Total do pedido: R$ 50,00

Pedido ID: 2
Data: 30/08/2025 22:01
Cliente ID: 1 | Nome: Erick Rezende | Telefone: 99999-1111
Itens:
Hambúrguer 2x | R$ 45,00
Total do pedido: R$ 45,00
```

Gerar Relatório Detalhado (Menu 9)
Título e data: imprime `=== Relatório Detalhado ===`. Para cada pedido, formata `pedido.tempo` com `"dd/MM/yyyy HH:mm"`.
Percorre todos os pedidos: para cada `Pedido p`:
Cabeçalho do pedido: mostra `Pedido ID` e **Data**.
Cliente: busca em `BancoDeDados.clientes` o registro cujo `id =p.idCliente` e imprime `ID | Nome | Telefone`.
Itens:
Para cada `Item` do cardápio, conta a **quantidade** desse item no pedido percorrendo `p.idItens` (cada ocorrência = 1 unidade).
Se `quantidade > 0`, calcula **subtotal = item.preco * quantidade**, imprime `nome quantidade x | R$ subtotal`, e acumula em `totalPedido`.
Total do pedido: imprime `Total do pedido: R$ <totalPedido>` (com 2 casas).

Saída

```
Escolha: 0
Saindo...

Process finished with exit code 0
```

Sair (Menu 0 – Sair)

Usuário digita **0**.
No `switch`, cai no `case 0` e imprime **"Saindo..."**.
Ao final da iteração, a condição do laço `do { ... } while (opcao != 0);` é avaliada como **falsa**, então o loop **termina**.
Fora do loop, o programa executa `sc.close()` (fecha o `Scanner`) e o método `main` termina — a aplicação **encerra**.
Observação: entradas inválidas caem no `catch`, onde `opcao = -1`; assim, **não sai** e o menu é mostrado novamente.

Referências

- BOOCH, G. Object-Oriented Analysis and Design with Applications. 3rd ed. Addison-Wesley, 2007.
- LARMAN, C. Applying UML and Patterns: An Introduction to Object- Oriented Analysis and Design and Iterative Development. 3rd ed. Prentice Hall, 2004.
- SOMMERVILLE, I. Software Engineering. 9th ed. Addison-Wesley, 2011.
- RUMBAUGH, J.; JACOBSON, I.; BOOCH, G. The Unified Modeling Language Reference Manual. 2nd ed. Addison-Wesley, 2005.
- W3SCHOOLS. Java ArrayList. Disponível em: https://www.w3schools.com/java/java_arraylist.asp. Acesso em: 30 ago. 2025.
- W3SCHOOLS. Java Date. Disponível em: https://www.w3schools.com/java/java_date.asp. Acesso em: 30 ago. 2025.