

## Desafio: Pedido de Venda

### 💡 Objetivo:

Desenvolver uma tela de Pedido de Venda com persistência em Firebird utilizando FireDAC, seguindo boas práticas de organização (ex.: MVC/Service/Repository) e Clean Code.

### 💡 Requisitos Funcionais:

1. Cliente
  - Operador informa Código do Cliente (não precisa cadastro).
  - Ao informar o código, exibir: Nome, Cidade, UF (somente leitura).
  - Validar cliente inexistente.
2. Itens do pedido (Produtos)
  - Para inserir item, informar:
    - o Código do Produto
    - o Quantidade
    - o Valor Unitário (pode vir preenchido com o preço de venda do produto e permitir edição)
  - Ao informar o código do produto, exibir Descrição (somente leitura).
  - Botão “Inserir/Atualizar Item” para incluir/atualizar no grid.
3. Grid de itens
  - Exibir colunas:
    - o Cód. Produto, Descrição, Quantidade, Vlr. Unitário, Vlr. Total (item)
  - Permitir produtos repetidos (itens em linhas distintas).
4. Edição e exclusão via teclado
  - Com foco no grid:
    - o ENTER: carregar item selecionado para os campos (produto/qtd/valor unitário) para edição.
  - Confirmar com o mesmo botão “Inserir/Atualizar Item”.
  - o DEL: excluir item com confirmação.
5. Total do pedido
  - Exibir no rodapé o Valor Total do Pedido (soma dos totais dos itens).
6. Gravar Pedido
  - Botão “Gravar Pedido” deve gravar:
    - o Cabeçalho do pedido
    - o Itens do pedido
  - Obrigatório usar transação (commit/rollback) e tratar erros.

## ✨Requisitos de Banco (Firebird)

### 1. Tabelas

CLIENTE

- CODIGO (PK), NOME, CIDADE, UF

PRODUTO

- CODIGO (PK), DESCRICAO, PRECO\_VENDA

PEDIDO

- NUMERO\_PEDIDO (PK), DATA\_EMISSAO, CODIGO\_CLIENTE (FK), VALOR\_TOTAL

PEDIDO\_ITEM

- ID (PK auto), NUMERO\_PEDIDO (FK), CODIGO\_PRODUTO (FK), QUANTIDADE, VLR\_UNITARIO, VLR\_TOTAL

### 2. Regras técnicas

- NUMERO\_PEDIDO deve ser sequencial crescente (Sequence/Generator no Firebird).
- PEDIDO\_ITEM.ID deve ser auto incremento (Identity/Generator).
- Criar FKs e índices mínimos:
  - o Índice em PEDIDO(CODIGO\_CLIENTE)
  - o Índice em PEDIDO\_ITEM(NUMERO\_PEDIDO)
  - o (Opcional) Índice em PEDIDO\_ITEM(CODIGO\_PRODUTO)

### 3. Carga de dados

- Popular CLIENTE e PRODUTO com 10 registros ou mais para teste.

## ✨Parte de Manutenção

Além do desenvolvimento, implemente uma pequena evolução + uma correção:

### 1. Evolução (simples)

- Adicionar no cabeçalho do pedido um campo “Observação” (texto curto) e persistir no banco.
  - o (Campo novo em PEDIDO: OBSERVACAO)

### 2. Correção (simples, mas importante)

- Garantir que ao excluir um item (DEL) o total do pedido seja recalculado corretamente e imediatamente atualizado na tela.

Observação: essa etapa é proposital para avaliar manutenção corretiva/evolutiva e cuidado com consistência.

## ✨Acesso ao Banco via INI (FireDAC)

Criar config.ini (na raiz do projeto) com:

- Database (caminho do .fdb)
- Username

- Password
- Server
- Port
- ClientLibrary (caminho do fbclient.dll)

A aplicação deve:

- Ler o INI e configurar a conexão FireDAC dinamicamente.
- Utilizar queries parametrizadas (evitar SQL Injection).
- Não utilizar componentes de terceiros.

Distribuição: incluir o fbclient.dll junto com a aplicação (ex.: pasta /bin).

### **Entregáveis**

No repositório (GitHub público):

1. Código fonte Delphi.
2. Script do banco (db.sql) contendo:
  - DDL (tabelas, sequences/generators, triggers se necessário, índices e FKs)
  - Inserts de cliente e produto
3. config.ini.example (exemplo sem “dados reais”).
4. README.md com:
  - Como criar o banco / rodar o script
  - Como configurar o INI
  - Como executar o projeto
  - Roteiro rápido de testes manuais (passo a passo)
5. Histórico de commits (versionamento):
  - Commits pequenos e descritivos (ex.: “feat: gravar pedido com transação”, “fix: recalcular total ao excluir item”, etc.)

### **Critérios de Avaliação**

- Delphi + FireDAC + Firebird (aderência à stack)
- Organização do código (camadas/responsabilidade única)
- Qualidade do SQL (joins, filtros, insert/update, uso de parâmetros)
- Tratamento de erros + transação (robustez)
- Performance básica (índices mínimos + queries corretas)
- Documentação (README + roteiro de testes)
- Boas práticas (segurança básica, clareza, versionamento)

Bônus (não obrigatório)

- Botão “Carregar Pedido” por número (traz cabeçalho + itens)
- Botão “Cancelar Pedido” (exclui pedido + itens com transação)
- Testes unitários (DUnitX) para regras de totalização