# 📋 Catálogo Completo de Funções e Recursos - HoStore

**Data**: Janeiro 2026 | **Versão**: 1.0.0 | **Linguagem**: Java 17+

## 📑 Índice

1. Funcionalidades por Módulo
2. Funções Principais por Classe
3. Fluxos de Negócio
4. Integração com APIs
5. Validações e Regras
6. Performance e Escalabilidade

## 🎯 Funcionalidades por Módulo

### 1️⃣ MÓDULO DE VENDAS (VendaService, VendaController)

#### 1.1 Criar e Gerenciar Vendas

- ☑ Criar nova venda (status: "aberta")
- ☑ Associar cliente obrigatoriamente
- ☑ Adicionar produtos ao carrinho dinamicamente
- ☑ Remover itens do carrinho
- ☑ Editar quantidade e preço antes de finalizar
- ☑ Aplicar desconto (percentual ou fixo)
- ☑ Consultar subtotal em tempo real
- ☑ Validar estoque disponível

#### 1.2 Finalizar Venda

- ☑ Confirmar itens e valores
- ☑ Selecionar forma de pagamento
- ☑ Configurar parcelamento (1-12x)
- ☑ Calcular juros automáticos
- ☑ Gerar comprovante PDF com QR code
- ☑ Imprimir comprovante direto
- ☑ Marcar status como "fechada"
- ☑ Atualizar estoque automaticamente

#### 1.3 Devoluções

- ☑ Processar devolução de produtos
- ☑ Reintegrar quantidade ao estoque

- ☑ Registrar motivo da devolução
- ☑ Manter histórico de devoluções
- ☑ Gerar nota de devolução

**1.4 Estornos**

- ☑ Estornar venda inteira (requer admin)
- ☑ Gerar venda negativa para auditoria
- ☑ Reverter movimentação de estoque
- ☑ Manter rastreamento completo
- ☑ Notificar auditoria

**1.5 Reabertura**

- ☑ Permitir reabertura apenas para admin
- ☑ Reverter status de "fechada" para "aberta"
- ☑ Manter histórico de modificações
- ☑ Exigir autenticação adicional

**1.6 Consulta e Filtros**

- ☑ Listar todas as vendas
- ☑ Filtrar por cliente
- ☑ Filtrar por período (data início/fim)
- ☑ Filtrar por valor total
- ☑ Filtrar por método de pagamento
- ☑ Filtrar por status (aberta/fechada/estornada)
- ☑ Busca textual integrada
- ☑ Paginação de resultados

**1.7 Análise e Relatórios**

- ☑ Total de vendas por período
- ☑ Ticket médio (valor total/quantidade)
- ☑ Vendas por cliente (ranking)
- ☑ Produtos mais vendidos
- ☑ Análise de margem
- ☑ Comparativo período anterior
- ☑ Exportar em PDF/Excel

---

## 2 MÓDULO DE ESTOQUE (EstoqueService, EstoqueDAO)

**2.1 Cadastro de Produtos (Múltiplas Categorias)**

**Cartas Individuais**

- ☑ Nome da carta
- ☑ Set/Coleção

- ☑ Número da carta
- ☑ Raridade
- ☑ Condição (Mint, NearMint, Excellent, etc.)
- ☑ Preço de custo
- ☑ Preço de venda
- ☑ Quantidade em estoque
- ☑ Imagem (opcional)

**Boosters**

- ☑ Nome do booster
- ☑ Set relacionado
- ☑ Quantidade de packs (1x, 3x, booster box)
- ☑ Preço unitário
- ☑ Quantidade em estoque

**Decks**

- ☑ Nome do deck
- ☑ Descrição/estratégia
- ☑ Lista completa de cartas (com link)
- ☑ Preço (montado)
- ☑ Quantidade em estoque

**ETBs (Elite Trainer Boxes)**

- ☑ Nome do produto
- ☑ Set específico
- ☑ Conteúdo descrito
- ☑ Preço
- ☑ Quantidade em estoque

**Acessórios**

- ☑ Nome do produto
- ☑ Marca
- ☑ Tipo (Sleeves, Playmat, dados, etc.)
- ☑ Cor/Variação
- ☑ Quantidade
- ☑ Preço

**Produtos Alimentícios**

- ☑ Nome
- ☑ Marca
- ☑ Categoria
- ☑ Data de validade
- ☑ Temperatura de armazenamento
- ☑ Preço
- ☑ Quantidade

### 2.2 Consultas e Buscas

- ☑ Busca por nome (autocomplete)
- ☑ Busca por categoria
- ☑ Busca por set/coleção
- ☑ Filtro por faixa de preço
- ☑ Filtro por estoque (tem/não tem/baixo)
- ☑ Busca combinada (múltiplos filtros)
- ☑ Ordenação por: nome, preço, estoque, data

### 2.3 Alertas Automáticos

- ☑ Estoque baixo (< 5 unidades)
- ☑ Sem estoque (0 unidades)
- ☑ Produtos próximos do vencimento (15 dias)
- ☑ Notificação em tempo real no dashboard
- ☑ Email opcional para admin

### 2.4 Movimentação de Estoque

- ☑ Entrada: Recebimento de compra
- ☑ Saída: Venda (automática)
- ☑ Ajuste: Perda/quebra manual
- ☑ Devolução: Produto devolvido
- ☑ Histórico completo com data/hora/usuário
- ☑ Rastreamento por número de série (opcional)

### 2.5 Pedidos de Compra

- ☑ Criar novo pedido
- ☑ Adicionar itens com quantidade e preço
- ☑ Selecionar fornecedor
- ☑ Acompanhar status (pendente/recebido/atrasado)
- ☑ Vincular com nota fiscal
- ☑ Registro de entrada automática

### 2.6 Dashboard de Estoque

- ☑ Total em unidades
- ☑ Total em valor (custo/venda)
- ☑ Quantidade de produtos cadastrados
- ☑ Produtos com estoque baixo (lista)
- ☑ PMZ (Preço Médio Ponderado)
- ☑ Valor de estoque por categoria
- ☑ Curva ABC

### 2.7 Exclusão e Auditoria

- ☑ Exclusão lógica (nunca apaga, marca como deletado)
- ☑ Registro de motivo de exclusão
- ☑ Histórico de todas as alterações
- ☑ Quem criou, quando, quem editou, quando

---

## 3 MÓDULO FINANCEIRO

### 3.1 Contas a Pagar (ContaPagarService)

- ☑ Registrar novo pagamento pendente
- ☑ Associar com fornecedor
- ☑ Data de vencimento
- ☑ Valor total
- ☑ Parcelar em múltiplas parcelas
- ☑ Juros e multa por atraso
- ☑ Registrar pagamento
- ☑ Gerar boleto (integração)
- ☑ Consultar por período
- ☑ Filtrar por status (pendente/pago/vencido)
- ☑ Relatório de fluxo

### 3.2 Contas a Receber (ContaReceberService)

- ☑ Criação automática de parcelas da venda
- ☑ Acompanhamento de recebiveis
- ☑ Simular recebimento antecipado
- ☑ Gerar cobrança automática
- ☑ Registrar cancelamento de parcela
- ☑ Consultar por cliente
- ☑ Filtrar por status (aberto/pago/vencido)
- ☑ Relatório de receivables

### 3.3 Crédito de Loja (CreditoLojaService)

- ☑ Gerenciar saldo de crédito por cliente
- ☑ Usar crédito para abater venda
- ☑ Recarregar crédito
- ☑ Histórico de movimentação
- ☑ Consultar saldo disponível
- ☑ Relatório de créditos em uso

### 3.4 Plano de Contas (PlanoContaService)

- ☑ Classificação de operações
- ☑ Integração com código NCM
- ☑ CSOSN (Código de Situação)
- ☑ CFOP (Código Fiscal de Operação)

- ☑ Natureza de operação
- ☑ Histórico de contas

**3.5 Relatórios Financeiros**

- ☑ Fluxo de caixa (projeção)
- ☑ Resultado do período (receita - despesa)
- ☑ Análise de pagamentos
- ☑ Contas a pagar por fornecedor
- ☑ Contas a receber por cliente
- ☑ Saldo de caixa
- ☑ Exportação em Excel/PDF

---

## 4 MÓDULO DE CLIENTES (ClienteService)

**4.1 Cadastro de Clientes**

- ☑ Nome/Razão Social
- ☑ CPF/CNPJ
- ☑ Email
- ☑ Telefone
- ☑ Data de nascimento
- ☑ Endereço completo
- ☑ Preferências de contato
- ☑ Limite de crédito

**4.2 Histórico de Compras**

- ☑ Listar todas as vendas do cliente
- ☑ Total gasto (acumulado)
- ☑ Última compra
- ☑ Frequência de compras
- ☑ Produtos favoritos

**4.3 Gestão de Crédito**

- ☑ Saldo atual
- ☑ Movimentações
- ☑ Débitos pendentes
- ☑ Histórico de pagamentos

**4.4 Filtros e Busca**

- ☑ Buscar por nome
- ☑ Buscar por CPF/CNPJ
- ☑ Filtrar por data de cadastro
- ☑ Filtrar por última compra

- ☑ Listar inativos

**4.5 Relatórios**

- ☑ Clientes com mais compras
- ☑ Novo clientes (mês/ano)
- ☑ Clientes em dia/atrasados
- ☑ Top 10 clientes
- ☑ Clientes inativos (> 90 dias)

---

## 5 MÓDULO FISCAL (DocumentoFiscalService)

**5.1 Tipos de Documentos**

- ☑ NFC-e (Nota Fiscal ao Consumidor)
- ☑ NFe (Nota Fiscal Eletrônica)
- ☑ CF (Cupom Fiscal)
- ☑ Nota de Devolução

**5.2 Configurações Fiscais**

- ☑ CNPJ da loja
- ☑ Inscrição Estadual
- ☑ Regime tributário
- ☑ Série da nota
- ☑ Numeração sequencial
- ☑ Certificado digital (se NFe)

**5.3 Campos Automáticos**

- ☑ Preenchimento de impostos (ICMS, IPI, PIS, COFINS)
- ☑ Cálculo de base tributária
- ☑ Código NCM (classificação)
- ☑ CSOSN (situação tributária)
- ☑ CFOP (operação)

**5.4 Geração de Documentos**

- ☑ Gerar XML compatível NF-e
- ☑ Assinar digitalmente (integração)
- ☑ Enviar para SEFAZ
- ☑ Registrar número de protocolo
- ☑ Armazenar XML para auditoria

**5.5 Comprovantes**

- ☑ PDF com dados completos
- ☑ QR code de validade

- ☑ Impressão térmica (80mm)
- ☑ Impressão A4
- ☑ Email para cliente

---

## 6 MÓDULO DE RELATÓRIOS (RelatoriosService)

### 6.1 Dashboard Principal

- ☑ Vendas do dia
- ☑ Estoque total (itens/valor)
- ☑ Clientes ativos
- ☑ Saldo de caixa
- ☑ Contas vencidas
- ☑ KPIs em tempo real

### 6.2 Relatórios de Vendas

- ☑ Vendas por dia/mês/ano/período
- ☑ Vendas por cliente
- ☑ Vendas por produto/categoria
- ☑ Ticket médio
- ☑ Produtos mais vendidos
- ☑ Análise de margem
- ☑ Comparativo com período anterior
- ☑ Variação (%)

### 6.3 Relatórios de Estoque

- ☑ Produtos em estoque
- ☑ Estoque por categoria
- ☑ Produtos baixos
- ☑ Produtos vencidos
- ☑ Movimentação (entrada/saída)
- ☑ Curva ABC
- ☑ Valor de estoque (custo/venda)
- ☑ PMZ por categoria

### 6.4 Relatórios Financeiros

- ☑ Fluxo de caixa
- ☑ Resultado do período
- ☑ Contas a pagar (por vencer/vencidas)
- ☑ Contas a receber (por vencer/vencidas)
- ☑ Saldo por cliente
- ☑ Análise de crédito
- ☑ Despesas por categoria

### 6.5 Exportação

- ☑ PDF com logo/cores
- ☑ Excel formatado (gráficos, tabelas)
- ☑ CSV para importação
- ☑ Impressão direta
- ☑ Envio por email

---

## 7 MÓDULO DE SISTEMA

### 7.1 Usuários e Autenticação

- ☑ Login com CPF/Email
- ☑ Senha encriptada (bcrypt)
- ☑ Recuperação de senha
- ☑ Alteração de senha
- ☑ Bloqueio após 3 tentativas
- ☑ Sessão com timeout

### 7.2 Permissões e Funções

- ☑ Admin: Acesso total
- ☑ Gerente: Vendas + Estoque + Relatórios
- ☑ Vendedor: Apenas vendas
- ☑ Estoquista: Apenas estoque
- ☑ Fiscal: Apenas documentos
- ☑ Roles customizáveis

### 7.3 Backup e Restauração

- ☑ Backup automático diário
- ☑ Backup manual sob demanda
- ☑ Compressão de dados (ZIP)
- ☑ Armazenamento em ./data/backup/
- ☑ Restauração completa
- ☑ Restauração parcial (por tabela)
- ☑ Sincronização em nuvem (futuro)

### 7.4 Auditoria e Logs

- ☑ Registra: QUEM fez QUÊ e QUANDO
- ☑ Log de login/logout
- ☑ Log de alterações em dados críticos
- ☑ Log de relatórios acessados
- ☑ Log de estornos/devoluções
- ☑ Exportar logs para análise
- ☑ Retenção por período configurável

### 7.5 Configurações

- ☑ Dados da loja (CNPJ, nome, endereço)
- ☑ Configurações de estoque (estoque mínimo, máximo)
- ☑ Configurações fiscais
- ☑ Configurações de pagamento
- ☑ Tema visual (light/dark)
- ☑ Impressora padrão
- ☑ Proxy para APIs

### 7.6 Sincronização de APIs

- ☑ Sincronizar Pokémon TCG
- ☑ Sincronizar Magic
- ☑ Sincronizar Yu-Gi-Oh!
- ☑ Sincronizar Digimon
- ☑ Sincronizar One Piece
- ☑ Agendamento automático
- ☑ Status da sincronização
- ☑ Cache local (offline)

---

## 🔧 Funções Principais por Classe

### VendaService

```
public class VendaService {
    // Criar e finalizar vendas
    public int finalizarVenda(VendaModel venda, List<VendaItemModel> itens)
    public VendaModel obterVenda(String vendaId)
    public List<VendaModel> listarVendas(LocalDate inicio, LocalDate fim)

    // Descontos
    public void aplicarDesconto(String vendaId, double desconto)
    public double calcularTotalComDesconto(VendaModel venda)

    // Devoluções
    public void processarDevolucao(String vendaId, String motivo)
    public void reintegrarEstoque(String vendaId)

    // Estornos
    public void estornarVenda(String vendaId, String motivo)
    public void reverterMovimentacao(String vendaId)

    // Relatórios
    public double getTotalVendasPeriodo(LocalDate inicio, LocalDate fim)
    public List<VendaModel> getVendasPorCliente(String clienteId)
    public Map<String, Integer> getTopProdutos(int quantidade)
}
```

### EstoqueService

```
public class EstoqueService {
    // Consultas
    public List<ProdutoEstoqueDTO> listarTodos()
    public ProdutoEstoqueDTO buscarPorId(String id)
    public List<ProdutoEstoqueDTO> buscarPorNome(String nome)
    public List<ProdutoEstoqueDTO> filtrarPorCategoria(String categoria)

    // Cadastro
    public void cadastrarProduto(ProdutoModel produto)
    public void atualizarProduto(ProdutoModel produto)
    public void deletarProduto(String id)

    // Movimentação
    public void entrada(String produtoId, int quantidade)
    public void saida(String produtoId, int quantidade)
    public void ajuste(String produtoId, int quantidade, String motivo)

    // Alertas
    public List<ProdutoEstoqueDTO> getEstoqueBaixo()
    public List<ProdutoEstoqueDTO> getVencidos()
    public List<ProdutoEstoqueDTO> getProximoVencimento(int dias)

    // Dashboard
    public double getTotalEstoqueValor()
    public Map<String, Double> getEstoquesPorCategoria()
    public double getPMZ()
}
```

### ClienteService

```
public class ClienteService {
    // CRUD
    public void cadastrarCliente(ClienteModel cliente)
    public ClienteModel obterCliente(String id)
    public List<ClienteModel> listarTodos()
    public void atualizarCliente(ClienteModel cliente)
    public void deletarCliente(String id)

    // Busca
    public ClienteModel buscarPorCpf(String cpf)
    public ClienteModel buscarPorEmail(String email)
    public List<ClienteModel> buscarPorNome(String nome)

    // Histórico
    public List<VendaModel> getHistoricoCompras(String clienteId)
    public double getTotalGasto(String clienteId)
    public LocalDate getUltimaCompra(String clienteId)
```

```
    // Crédito
    public double getSaldoCredito(String clienteId)
    public void adicionarCredito(String clienteId, double valor)
    public void deduzirCredito(String clienteId, double valor)
}
```

### DocumentoFiscalService

```
public class DocumentoFiscalService {
    // Emissão
    public void emitirNFCe(VendaModel venda)
    public void emitirNFe(VendaModel venda)
    public void emitirNotaDevolucao(VendaDevolucaoModel devolucao)

    // Validação
    public boolean validarDadosFiscais(VendaModel venda)
    public boolean validarNCM(String ncm)
    public boolean validarCFOP(String cfop)

    // Geração
    public String gerarXML(VendaModel venda)
    public void assinarDigitalmente(String xmlPath)
    public void enviarParaSEFAZ(String xml)

    // Comprovante
    public void gerarPDF(VendaModel venda, String caminho)
    public void imprimirComprovante(VendaModel venda)
    public void enviarPorEmail(VendaModel venda, String email)
}
```

### RelatoriosService

```
public class RelatoriosService {
    // Dashboard
    public DashboardHomeModel getDashboardPrincipal()
    public DashboardKpisModel getKPIs()

    // Vendas
    public List<VendaModel> getRelatorioVendas(LocalDate inicio, LocalDate fim)
    public Map<String, Double> getVendasPorCliente(LocalDate inicio, LocalDate
fim)
    public Map<String, Integer> getVendasPorProduto(LocalDate inicio, LocalDate
fim)
    public double getTicketMedio(LocalDate inicio, LocalDate fim)

    // Estoque
    public List<EstoqueModel> getRelatorioEstoque()
    public List<ProdutoEstoqueDTO> getEstoqueBaixo()
    public Map<String, Double> getEstoquesPorCategoria()
```

```
    // Financeiro
    public FluxoCaixaModel getFluxoCaixa(LocalDate inicio, LocalDate fim)
    public ResultadoModel getResultadoPeriodo(LocalDate inicio, LocalDate fim)

    // Exportação
    public void exportarPDF(RelatorioModel relatorio, String caminho)
    public void exportarExcel(RelatorioModel relatorio, String caminho)
    public void exportarCSV(RelatorioModel relatorio, String caminho)
}
```

---

## 📊 Fluxos de Negócio

### Fluxo 1: Nova Venda (Passo a Passo)

```
1. PainelVendas.java (UI)
   └─ Botão "Nova Venda"
      └─ VendaNovaDialog.java abre

2. VendaNovaDialog (Carrinho)
   ├─ SelectClienteDialog (Seleciona cliente)
   ├─ SelectProdutoDialog (Busca produto)
   │  └─ EstoqueService.buscarPorNome()
   ├─ Adicionar ao carrinho (VendaItemModel)
   ├─ Aplicar desconto
   │  └─ VendaController.aplicarDesconto()
   └─ Botão "Finalizar"

3. VendaFinalizarDialog (Confirmação)
   ├─ Revisar itens
   ├─ ParcelamentoDialog
   │  └─ Configura 1-12x
   ├─ SelectPagamento (dinheiro/cartão/PIX)
   └─ Botão "Confirmar"

4. VendaService.finalizarVenda()
   ├─ Validar dados
   ├─ Criar registro VendaModel
   ├─ Inserir itens (VendaItemModel)
   ├─ Baixar estoque automaticamente
   │  └─ EstoqueService.saida()
   ├─ Registrar pagamento
   │  └─ DocumentoFiscalService.emitirNFCe()
   └─ Gerar comprovante PDF
      └─ PDFGenerator.gerarPDF()

5. ComprovanteFiscalDialog
   ├─ Exibir PDF
   ├─ Opção Imprimir
   └─ Venda finalizada ☑
```

### Fluxo 2: Novo Produto no Estoque

```
1. PainelEstoque.java
   └─ Botão "Novo Item"
      └─ SelecionarCategoriaDialog

2. SelecionarCategoriaDialog
   ├─ Carta
   │  └─ CadastroCartaDialog
   │     ├─ Nome, Set, Número, Raridade, Condição
   │     └─ Preço custo/venda
   ├─ Booster
   │  └─ CadastroBoosterDialog
   ├─ Deck
   │  └─ CadastroDeckDialog
   ├─ ETB
   │  └─ CadastroEtbDialog
   ├─ Acessório
   │  └─ CadastroAcessorioDialog
   └─ Alimentício
      └─ CadastroProdutoAlimenticioDialog

3. Cada Dialog
   ├─ Preenche dados específicos
   ├─ Validação obrigatória
   └─ Salva no banco

4. EstoqueService.cadastrarProduto()
   ├─ Criar registro com ID único
   ├─ Inserir em tabela específica
   ├─ Atualizar tabela estoque (view)
   └─ Registrar em auditoria

5. PainelEstoque atualiza
   └─ Produto aparece na lista ☑
```

### Fluxo 3: Entrada de Produtos (Recebimento de Compra)

```
1. PainelEstoque
   └─ Botão "Entrada de Produtos"
      └─ EntradaProdutosDialog

2. EntradaProdutosDialog
   ├─ Selecionar Pedido de Compra (ou novo)
   │  └─ NovoPedidoEstoqueDialog
   ├─ Listar produtos do pedido
   ├─ Confirmar quantidades
   └─ Vincular Nota Fiscal (arquivo PDF/XML)
```

```
3. Validação
   ├─ Verificar produto existe
   ├─ Verificar quantidade
   └─ Verificar nota fiscal

4. MovimentacaoEstoqueService
   ├─ Registrar entrada
   │  └─ MovimentacaoEstoqueDAO.insert()
   ├─ Atualizar quantidade
   │  └─ EstoqueDAO.update()
   └─ Atualizar tabela estoque (view)

5. Banco de Dados
   ├─ INSERT movimentacao_estoque
   ├─ UPDATE estoque
   └─ Auditoria registrada

6. Painel atualiza
   └─ Estoque reflete entrada ☑
```

---

## 🌐 Integração com APIs

### 1. Pokémon TCG API

```
// PokeTcgApi.java
public static String listarSetsPokemon() {
    // GET https://api.pokemontcg.io/v2/sets
    // Cache: ./data/cache/pokemontcg_sets.json
    // Retorna: JSON com todos os sets disponíveis
}

public static String listarCardsPorSet(String setId) {
    // GET https://api.pokemontcg.io/v2/cards?q=set.id:sv01
    // Cache: ./data/cache/pokemontcg_cards_[setId].json
    // Retorna: JSON com cartas do set
}
```

**Sincronização**:

- Agendada diariamente (02:00 AM)
- Ou manual via Menu → Ajustes → Sincronizar TCG
- Fallback automático (cache) se offline

### 2. Magic: The Gathering (Scryfall)

```
// CardGamesApi.java
public static String listarSetsMagic() {
    // GET https://api.scryfall.com/sets
```

```
    // Cache: ./data/cache/magic_sets.json
}

public static String listarCardsMagicPorSet(String setCode) {
    // GET https://api.scryfall.com/cards/search?q=set:[code]
    // Paginação: 175 cartas por página
}
```

### 3. Yu-Gi-Oh! (YGOPRODeck)

```
// CardGamesApi.java
public static String listarSetsYgo() {
    // GET https://db.ygoprodeck.com/api/v7/cardsets.php
    // Cache: ./data/cache/yugioh_sets.json
}

public static String listarCardsYgoPorSet(String setName) {
    // GET https://db.ygoprodeck.com/api/v7/cardinfo.php?set=[name]
}
```

### 4. Digimon Card Game

```
public static String listarCardsDigi() {
    // GET https://digimoncard.io/api-public/getAllCards
    // Cache: ./data/cache/digimon_all_cards.json
    // Retorna: Lista completa de cartas
}
```

### 5. One Piece TCG

```
public static String listarSetsOnepiece() {
    // GET https://optcgapi.com/api/allSets/
    // Cache: ./data/cache/onepiece_sets.json
}

public static String listarCardsOnepiecePorSet(String setCode) {
    // GET https://optcgapi.com/api/allSetCards/[code]/
}
```

**Rate Limiting**:

- Max 30 requisições por minuto
- Timeout: 15 segundos por request
- Retry: 3 tentativas automáticas
- Backoff exponencial se limite atingido

# ☑ Validações e Regras

## Vendas

| Regra | Status | Descrição |
|---|---|---|
| Cliente obrigatório | ☑ Ativo | Não finalizando venda sem cliente |
| Mínimo 1 item | ☑ Ativo | Não permitir venda vazia |
| Estoque disponível | ☑ Ativo | Validar quantidade antes de vender |
| Preço > 0 | ☑ Ativo | Preço não pode ser negativo |
| Total > 0 | ☑ Ativo | Total deve ser positivo |
| Desconto ≤ 100% | ☑ Ativo | Desconto não pode exceder 100% |
| Imprimir comprovante | ☑ Ativo | Obrigatório gerar comprovante |
| Auditoria | ☑ Ativo | Registrar quem, quando, o quê |

## Estoque

| Regra | Status | Descrição |
|---|---|---|
| Produto único | ☑ Ativo | Não permitir duplicação |
| Nome obrigatório | ☑ Ativo | Campo nome não pode ser vazio |
| Preço ≥ 0 | ☑ Ativo | Preço não negativo |
| Quantidade ≥ 0 | ☑ Ativo | Quantidade não negativa |
| Categoria válida | ☑ Ativo | Apenas categorias pré-definidas |
| Alertas automáticos | ☑ Ativo | Estoque baixo (<5), vencidos |
| Histórico completo | ☑ Ativo | Rastrear todas as mudanças |

## Financeiro

| Regra | Status | Descrição |
|---|---|---|
| Vencimento válido | ☑ Ativo | Data deve ser no futuro |
| Valor > 0 | ☑ Ativo | Valores negativos não permitidos |
| Juros ≥ 0 | ☑ Ativo | Juros não podem ser negativos |
| Parcelas 1-12 | ☑ Ativo | Limitar parcelamento |
| Intervalo mínimo | ☑ Ativo | 30 dias entre parcelas |
| Saldo válido | ☑ Ativo | Não usar crédito maior que saldo |

### Fiscal

| Regra | Status | Descrição |
|---|---|---|
| CNPJ válido | ☑ Ativo | Formato XX.XXX.XXX/XXXX-XX |
| CPF válido | ☑ Ativo | Validar dígitos verificadores |
| NCM válido | ☑ Ativo | 8 dígitos numéricos |
| CFOP válido | ☑ Ativo | Código fiscal válido |
| CSOSN válido | ☑ Ativo | Situação tributária |

---

## ⚡ Performance e Escalabilidade

### Índices de Banco de Dados

```
-- Tabelas críticas
CREATE INDEX idx_vendas_cliente ON vendas(cliente_id);
CREATE INDEX idx_vendas_data ON vendas(data_venda);
CREATE INDEX idx_vendas_status ON vendas(status);

CREATE INDEX idx_estoque_categoria ON estoque(categoria);
CREATE INDEX idx_estoque_nome ON estoque(nome);

CREATE INDEX idx_clientes_cpf ON clientes(cpf);
CREATE INDEX idx_clientes_nome ON clientes(nome);

CREATE INDEX idx_movimentacao_data ON movimentacao_estoque(data);
```

### Cache Strategy

- **L1 Cache**: Memória (HashMap) - sesão do usuário
- **L2 Cache**: SQLite com índices - banco local
- **L3 Cache**: Arquivo JSON - APIs externas
- **TTL**: 24 horas para dados de API

### Limite de Dados

| Conceito | Limite | Sugestão |
|---|---|---|
| Produtos | Sem limite | ~50k testado |
| Vendas | Sem limite | Arquivar >1 ano |
| Clientes | Sem limite | ~10k ideal |
| Usuários | 100 | Ideal 5-20 |
| Relatórios | Períodos de 90 dias | Quebrar em trimestres |

---

**Documentação atualizada**: Janeiro 2026 | **Versão**: 1.0.0

---

## Links Úteis

- README Completo
- Estoque.md
- Vendas.md
- GitHub