

Implementação Completa de NFe/NFCe no HoStore

Data: Janeiro 2026

Status: Guia Técnico - Implementação Necessária

Versão: 1.0

Índice

1. [O que é NFe e NFCe?](#)
 2. [O que você precisa ter \(Requisitos\)](#)
 3. [Análise do projeto atual](#)
 4. [Arquitetura da solução](#)
 5. [Tabelas fiscais necessárias](#)
 6. [Cálculo de impostos](#)
 7. [Guia de implementação passo a passo](#)
 8. [Fluxo de emissão](#)
 9. [Tratamento de erros e retorno](#)
 10. [Testes e validações](#)
-

O que é NFe e NFCe?

NFe (Nota Fiscal Eletrônica)

- **Sigla:** NF-e
- **Código Modelo:** 55
- **Uso:** Operações B2B (Pessoa Jurídica para Pessoa Jurídica)
- **Quando usar:** Venda para revendedor, distribuidor ou outra empresa
- **Exigências:**
 - CPF/CNPJ do cliente
 - Certificado digital (A1 ou A3)
 - SEFAZ integração via webservice

NFCe (Nota Fiscal do Consumidor Eletrônica)

- **Sigla:** NFC-e
- **Código Modelo:** 65
- **Uso:** Operações B2C (Varejo para Consumidor Final)
- **Quando usar:** Venda em balcão de loja
- **Exigências:**
 - CPF ou CNPJ do cliente (pode ser vazio com "Consumidor" genérico)
 - CSC (Código de Segurança do Contribuinte)
 - Certificado digital (opcional para alguns estados)
 - Emissão via webservice simplificado

Diferenças Principais

Aspecto	NFe	NFCe
Código Modelo	55	65
Ambiente	SEFAZ Nacional	SEFAZ estadual
Certificado	Obrigatório (A1)	CSC ou Certificado
Cliente	B2B (Obrigatório)	B2C (Opcional/Genérico)
Atributos	Completos	Simplificados
Contingência	Yes-Manoel/SVC	Offline (SAT)
Validade XML	30 dias	24 horas

📋 O que você precisa ter (Requisitos)

1. Requisitos Legais e Administrativos

Para NFe

- CNPJ da empresa registrado na RF B (Receita Federal)
- Inscrição Estadual ativa
- Regime tributário definido (Simples Nacional, Lucro Presumido, etc.)
- Habilitação para emitir NFe na UF (validar em SEFAZ)
- Certificado digital tipo A1 (1 ano) ou A3 (3 anos)

Para NFCe

- CNPJ da empresa registrado
- Inscrição Estadual ativa (alguns estados exigem)
- Regime tributário definido
- Código de Segurança do Contribuinte (CSC) - gerado no portal da SEFAZ
- Certificado digital (varável por estado)

2. Certificado Digital

Tipo A1 (Recomendado para iniciantes)

- **Formato:** Arquivo .PFX ou .P12 com senha
- **Duração:** 1 ano
- **Armazenamento:** No computador/servidor
- **Custo:** R\$ 100-300/ano
- **Emissão:** AC Raiz (Serasa, Certisign, Comodo, etc.)
- **Como obter:** Solicitar à empresa certificadora

Tipo A3

- **Formato:** Token ou SmartCard
- **Duração:** 3 anos
- **Armazenamento:** Dispositivo criptográfico
- **Custo:** R\$ 200-500/3 anos
- **Mais seguro:** Recomendado para produção

3. CSC - Código de Segurança do Contribuinte

Para NFCe você precisa de:

- **CSC (Código de Segurança):** String de até 36 caracteres gerada no portal SEFAZ
- **idCSC (ID do CSC):** Número de 1 a 65535 que identifica qual CSC usar

Como obter:

1. Acessar portal da SEFAZ do seu estado
2. Aba "NFCe" → "Meu CSC"
3. Gerar novo CSC (criptografia)
4. Guardar com segurança

4. APIs e Webservices

Para NFe

- **SEFAZ Nacional:** <https://nfe.fazenda.gov.br/webservices/nfeautorizacao4/nfeautorizacao4.asmx>
- **SEFAZ Estadual:** Varia conforme UF
- **Ambiente de Testes:** <https://homolog.sefaz.fazenda.gov.br/>

Para NFCe

- **SEFAZ Nacional:** <https://nfe.fazenda.gov.br/webservices/nfeconsultacao/nfeconsultacao4.asmx>
- **SEFAZ Estadual:** Webservice simplificado
- **Ambiente de Testes:** <https://homolog.nfce.sefaz.fazenda.gov.br/>

Análise do Projeto Atual

O que JÁ existe no HoStore

Tabelas Fiscais Base ([util/DB.java](#)):

```
CREATE TABLE ncm (
    codigo TEXT PRIMARY KEY,
    descricao TEXT NOT NULL
);

CREATE TABLE cfop (
    codigo TEXT PRIMARY KEY,
```

```

descricao TEXT NOT NULL
);

CREATE TABLE csosn (
  codigo TEXT PRIMARY KEY,
  descricao TEXT NOT NULL
);

CREATE TABLE origem (
  codigo TEXT PRIMARY KEY,
  descricao TEXT NOT NULL
);

CREATE TABLE unidades (
  codigo TEXT PRIMARY KEY,
  descricao TEXT NOT NULL
);

```

Tabelas de Documentos Fiscais:

```

CREATE TABLE sequencias_fiscais (
  id TEXT PRIMARY KEY,
  modelo TEXT NOT NULL,          -- "NFe" ou "NFSe"
  codigo_modelo INTEGER NOT NULL, -- 55 ou 65
  serie INTEGER NOT NULL,
  ambiente TEXT NOT NULL,        -- "producao" ou "homologacao"
  ultimo_numero INTEGER NOT NULL DEFAULT 0
);

CREATE TABLE documentos_fiscais (
  id TEXT PRIMARY KEY,
  venda_id INTEGER NOT NULL,
  modelo TEXT NOT NULL,
  codigo_modelo INTEGER NOT NULL,
  serie INTEGER NOT NULL,
  numero INTEGER NOT NULL,
  ambiente TEXT NOT NULL,
  status TEXT NOT NULL,          -- "pendente", "autorizada", "cancelada"
  chave_acesso TEXT,
  protocolo TEXT,
  recibo TEXT,
  xml TEXT,
  erro TEXT,
  total_produtos REAL,
  total_desconto REAL,
  total_acrescimo REAL,
  total_final REAL
);

CREATE TABLE documentos_fiscais_itens (
  id INTEGER PRIMARY KEY,
  documento_id TEXT NOT NULL,

```

```

venda_item_id INTEGER,
produto_id TEXT,
descricao TEXT NOT NULL,
ncm TEXT NOT NULL,
cfop TEXT NOT NULL,
csosn TEXT NOT NULL,
origem TEXT NOT NULL,
unidade TEXT NOT NULL,
quantidade INTEGER NOT NULL,
valor_unit REAL NOT NULL,
desconto REAL DEFAULT 0,
acrescimo REAL DEFAULT 0,
total_item REAL NOT NULL
);

CREATE TABLE documentos_fiscais_pagamentos (
  id INTEGER PRIMARY KEY,
  documento_id TEXT NOT NULL,
  tipo TEXT NOT NULL,          -- "01" (dinheiro), "02" (cheque), etc.
  valor REAL NOT NULL
);

```

Campos fiscais em Produtos:

```

"ncm TEXT,"           // ex: "95049090"
"cfop TEXT,"          // ex: "5102"
"csosn TEXT,"         // ex: "102"
"origem TEXT,"        // ex: "0"
"unidade TEXT"        // ex: "UN"

```

Configuração da Loja (config_loja):

```

cnpj TEXT NOT NULL,
inscricao_estadual TEXT,
regime_tributario TEXT,
cnae TEXT,
endereco_logradouro TEXT,
endereco_numero TEXT,
endereco_complemento TEXT,
endereco_bairro TEXT,
endereco_municipio TEXT,
endereco_uf TEXT,
endereco_cep TEXT,
telefone TEXT,
email TEXT,
-- (e mais campos de configuração)

```

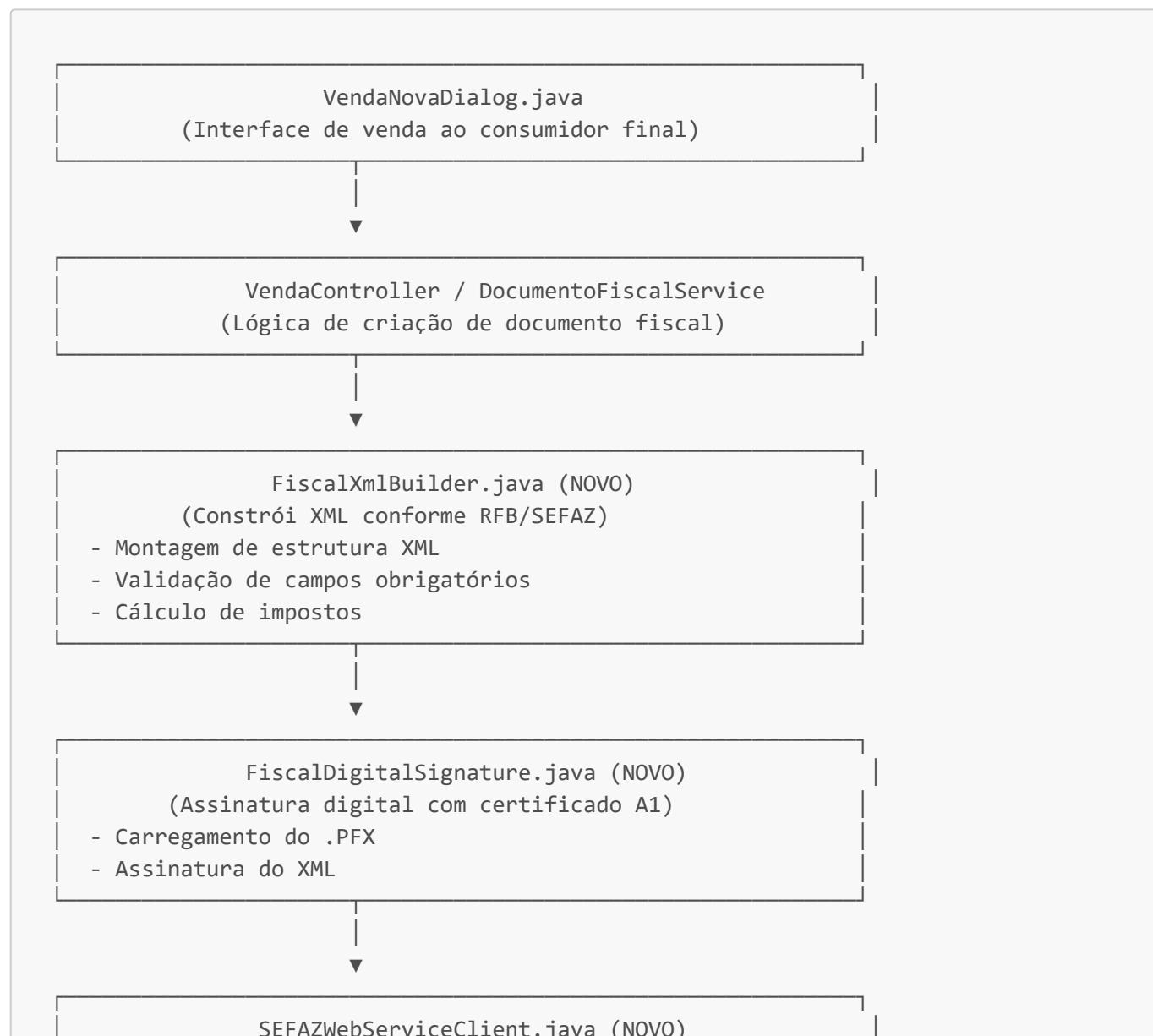
Serviços existentes:

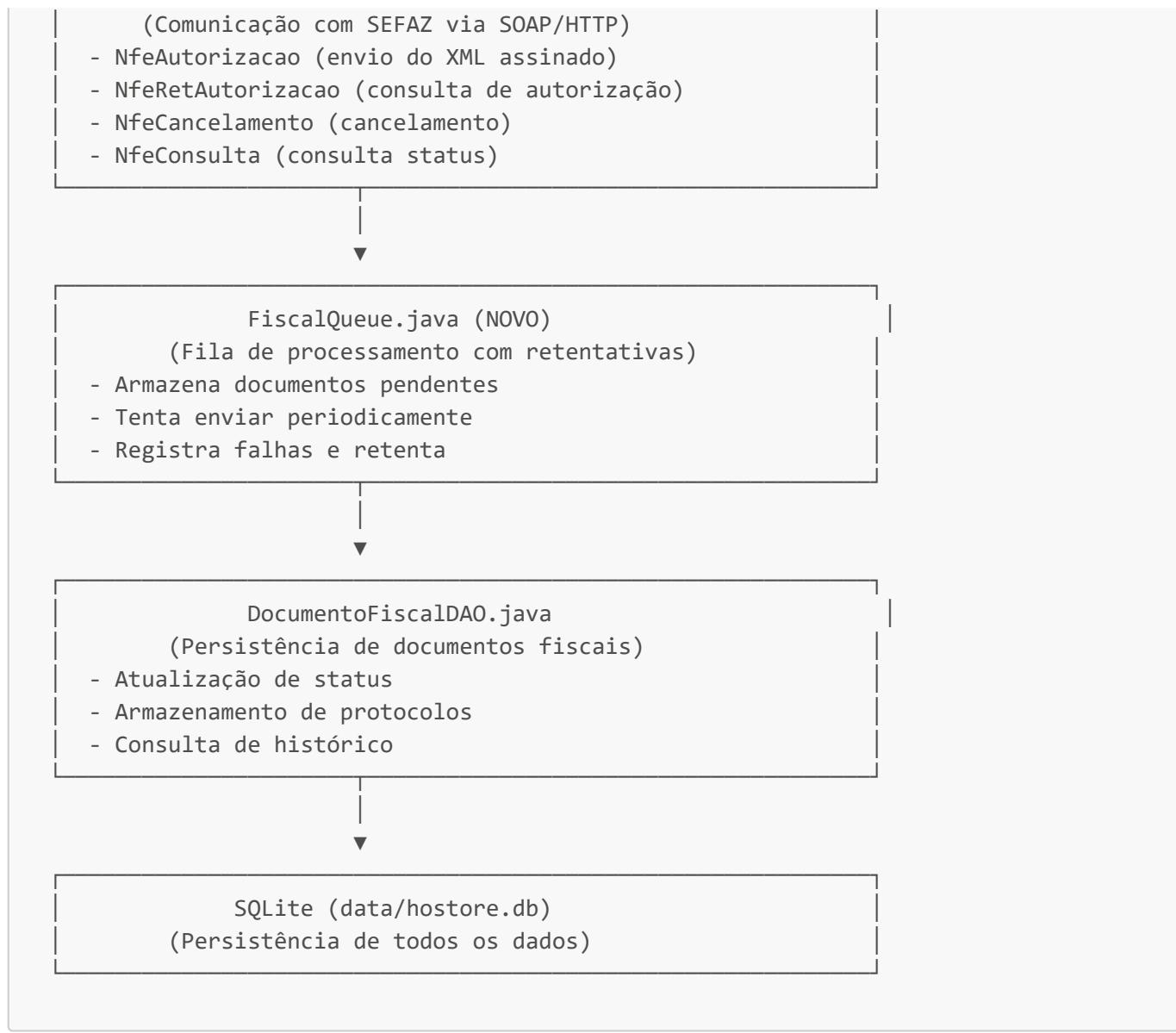
- `Fiscal ApiService.java`: Sincronização de dados fiscais
- `DocumentoFiscalService.java`: Criação de documentos pendentes
- `FiscalCatalogDAO.java`: Acesso aos dados fiscais

O que FALTA

- ✗ **Integração com SEFAZ** (geração, assinatura digital, envio)
- ✗ **Biblioteca de XSD/XML** (construção do XML correto)
- ✗ **Validação de regras fiscais** (ICMS, IPI, PIS/COFINS)
- ✗ **Comunicação com Certificado Digital**
- ✗ **Fila de envio e retentativa**
- ✗ **Contingência (Yes-Manoel)**
- ✗ **Consulta de autorização/cancelamento**
- ✗ **Geração de DANFE** (gráfico NF-e)
- ✗ **Configuração de Regime Tributário**
- ✗ **Cálculo automático de impostos por produto**

█ Arquitetura da Solução





Tabelas Fiscais Necessárias

1. Tabelas que JÁ existem

ncm - Nomenclatura Comum do MERCOSUL

```

CREATE TABLE ncm (
  codigo TEXT PRIMARY KEY,          -- ex: "95049090"
  descricao TEXT NOT NULL          -- ex: "Cartas de jogar"
);
  
```

Dados essenciais por categoria:

Categoria	NCM	Descrição
Cartas colecionáveis	95049090	Cartas de jogar (Magic, Yu-Gi-Oh, etc.)
Pokémon TCG	95049090	Cartas Pokémon

Categoría	NCM	Descrição
Boosters	95049090	Pacotes de cartas
Acessórios (sleeves)	39229000	Embalagens plásticas
Playmats	59089000	Tapetes

cfop - Código Fiscal de Operações e Prestações

```
CREATE TABLE cfop (
    codigo TEXT PRIMARY KEY,          -- ex: "5102"
    descricao TEXT NOT NULL          -- ex: "Venda de mercadoria"
);
```

CFOPs mais usados para varejo:

CFOP	Descrição
5102	Venda de mercadoria adquirida ou recebida de terceiros
5101	Venda de mercadoria de produção própria
1102	Compra para industrialização (se comprar de fornecedor)

csosn - Código de Situação da Operação Negocial

```
CREATE TABLE csosn (
    codigo TEXT PRIMARY KEY,          -- ex: "102"
    descricao TEXT NOT NULL          -- ex: "Tributada pelo Simples Nacional com
                                     permissão de crédito"
);
```

CSOSNs para Simples Nacional (mais comum):

CSOSN	Descrição	ICMS	PIS	COFINS
100	Tributada pelo Simples Nacional - ICMS	Sim	Sim	Sim
101	Tributada pelo Simples Nacional - ICMS (alíquota diferenciada)	Sim	Sim	Sim
102	Tributada pelo Simples Nacional sem permissão de crédito	Não	Não	Não
103	Isenção do ICMS do Simples Nacional para faixas de receita bruta	Não	Sim	Sim
201	Tributada pelo Simples Nacional (Lei Complementar 123/2006)	Sim	Sim	Sim
202	Tributada pelo Simples Nacional (alíquota diferenciada)	Sim	Sim	Sim
203	Isenção do ICMS do Simples Nacional para faixas de receita	Não	Sim	Sim

CSOSN	Descrição	ICMS	PIS	COFINS
300	Imune/Isenta	Não	Não	Não
400	Isenta	Não	Não	Não
500	ICMS cobrado anteriormente por substituição tributária	Não	Não	Não

origem - Origem da Mercadoria

```
CREATE TABLE origem (
    codigo TEXT PRIMARY KEY,          -- ex: "0"
    descricao TEXT NOT NULL          -- ex: "Nacional"
);
```

Código Descrição

0	Nacional
1	Estrangeira - importação direta
2	Estrangeira - adquirida no mercado interno
3	Nacional com adição de produto importado
4	Nacional com valor agregado de produto importado

unidades - Unidades de Medida

```
CREATE TABLE unidades (
    codigo TEXT PRIMARY KEY,          -- ex: "UN"
    descricao TEXT NOT NULL          -- ex: "Unidade"
);
```

2. Novas tabelas necessárias ✕ (CRIAR)

T1: imposto_icms - Configurações de ICMS por Estado

```
CREATE TABLE imposto_icms (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    estado TEXT NOT NULL,           -- UF de origem (SP, RJ, MG...)
    estado_destino TEXT NOT NULL,   -- UF de destino
    ncm TEXT NOT NULL,             -- Código NCM
    aliquota_consumidor REAL,      -- Alíquota para consumidor (NFSe)
    aliquota_contribuinte REAL,     -- Alíquota para empresa (NFe)
    reducao_base REAL DEFAULT 0,   -- Redução de base (%)
    mva_bc REAL DEFAULT 0,          -- Margem de valor agregado
    ativo INTEGER DEFAULT 1,
```

```

FOREIGN KEY(ncm) REFERENCES ncm(codigo),
UNIQUE(estado, estado_destino, ncm)
);

```

T2: imposto_ipi - Configurações de IPI

```

CREATE TABLE imposto_ipi (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    ncm TEXT NOT NULL,
    aliquota REAL, -- Alíquota (%)
    cnpj_produtor TEXT, -- CNPJ do produtor (opcional)
    ativo INTEGER DEFAULT 1,
    FOREIGN KEY(ncm) REFERENCES ncm(codigo),
    UNIQUE(ncm, cnpj_produtor)
);

```

T3: imposto_pis_cofins - Configurações de PIS/COFINS

```

CREATE TABLE imposto_pis_cofins (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    ncm TEXT NOT NULL,
    cst_pis TEXT, -- ex: "01", "02", "04"
    aliquota_pis REAL, -- Alíquota PIS (%)
    cst_cofins TEXT, -- ex: "01", "02", "04"
    aliquota_cofins REAL, -- Alíquota COFINS (%)
    ativo INTEGER DEFAULT 1,
    FOREIGN KEY(ncm) REFERENCES ncm(codigo),
    UNIQUE(ncm, cst_pis, cst_cofins)
);

```

T4: configuracao_nfe_nfce - Configuração geral NFe/NFCe

```

CREATE TABLE configuracao_nfe_nfce (
    id TEXT PRIMARY KEY,
    -- NFe
    emitir_nfe INTEGER DEFAULT 0, -- 1 = ativar
    certificado_path_nfe TEXT, -- Caminho do .pfx
    certificado_senha_nfe TEXT, -- Senha (criptografada)
    serie_nfe INTEGER DEFAULT 1,
    numero_inicial_nfe INTEGER DEFAULT 1,
    -- NFCe
    emitir_nfce INTEGER DEFAULT 1, -- 1 = ativar
    csc_nfce TEXT, -- Código de Segurança
    id_csc_nfce INTEGER, -- ID do CSC
    certificado_path_nfce TEXT, -- Path (se necessário)
);

```

```

certificado_senha_nfce TEXT,          -- Senha
serie_nfce INTEGER DEFAULT 1,
numero_inicial_nfce INTEGER DEFAULT 1,

-- Geral
ambiente TEXT DEFAULT 'homologacao', -- "homologacao" ou "producao"
regime_tributario TEXT,              -- "SN" (Simples Nacional), "LP", "LL"

-- Contingência
uso_contingencia INTEGER DEFAULT 0, -- 1 = ativar modo contingência
tipo_contingencia TEXT,            -- "SVC", "SVC-AN", "OFFLINE"

-- WebService
url_webservice_nfe TEXT,
url_webservice_nfce TEXT,
timeout_segundos INTEGER DEFAULT 30,

-- Impressão
nome_impressora TEXT,
largura_papel INTEGER DEFAULT 80, -- Largura da NFCe em mm

criado_em TEXT,
alterado_em TEXT,
alterado_por TEXT
);

```

T5: fila_emissao_nf - Fila de emissão

```

CREATE TABLE fila_emissao_nf (
    id TEXT PRIMARY KEY,
    documento_id TEXT NOT NULL,      -- FK para documentos_fiscais
    tipo TEXT NOT NULL,              -- "NFe" ou "NFCe"
    status TEXT DEFAULT 'pendente',   -- "pendente", "enviando", "autorizada",
    "erro"
    tentativas INTEGER DEFAULT 0,
    max_tentativas INTEGER DEFAULT 5,
    proximo_envio TEXT,             -- Quando tentar novamente
    mensagem_erro TEXT,
    criado_em TEXT,
    alterado_em TEXT,
    FOREIGN KEY(documento_id) REFERENCES documentos_fiscais(id)
);

```

T6: Log de emissão fiscal

```

CREATE TABLE log_emissao_fiscal (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    documento_id TEXT,

```

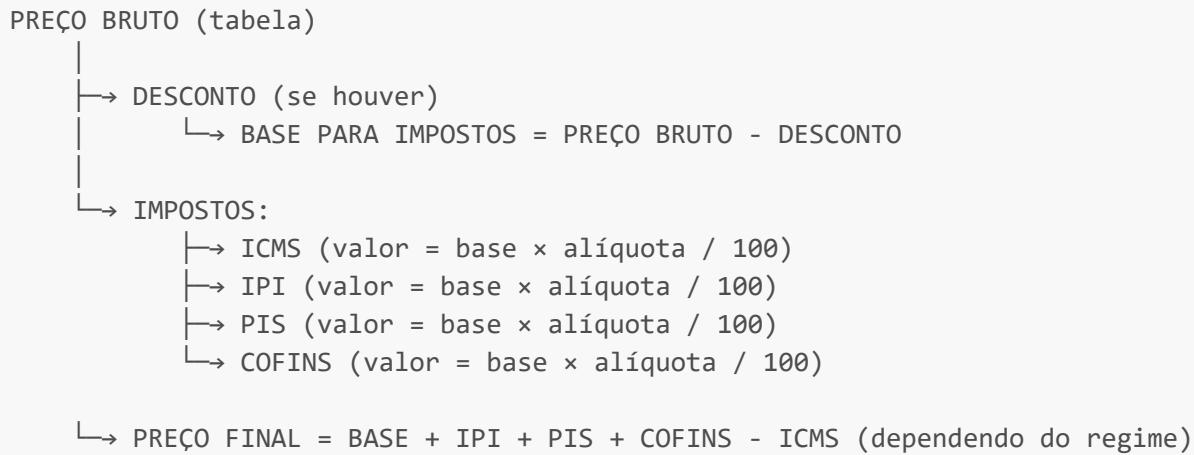
```

    tipo_operacao TEXT,           -- "envio", "consulta", "cancelamento"
    status_anter TEXT,
    status_depois TEXT,
    resposta_sefaz TEXT,         -- XML de resposta
    tempo_execucao_ms INTEGER,
    erro TEXT,
    criado_em TEXT,
    criado_por TEXT,
    FOREIGN KEY(documento_id) REFERENCES documentos_fiscais(id)
);

```

⌚ Cálculo de Impostos

1. Estrutura de Cálculo



2. Exemplo Prático

Produto: Booster Magic: The Gathering

Preço unitário: R\$ 50,00

Quantidade: 2

Base de cálculo: R\$ 100,00

Para Simples Nacional (mais comum em varejo)

Base: R\$ 100,00

ICMS: Base × alíquota / 100

- Alíquota ICMS SN: ~7% (varia por estado)
- Valor ICMS: $100 \times 7 / 100 = R\$ 7,00$

PIS: Base × alíquota / 100

- Alíquota PIS SN: ~1,25%
- Valor PIS: $100 \times 1,25 / 100 = R\$ 1,25$

COFINS: Base × alíquota / 100
- Alíquota COFINS SN: ~5,75%
- Valor COFINS: $100 \times 5,75 / 100 = R\$ 5,75$

IPI: Geralmente 0% em cartas (verificar NCM específico)

TOTAL IMPOSTOS: $7,00 + 1,25 + 5,75 = R\$ 14,00$

PREÇO FINAL (NFCe): R\$ 100,00 (já incluso no preço final)

Nota: Em Simples Nacional, o imposto já está dentro do preço de venda

3. Cálculo por Regime Tributário

A. Simples Nacional (SN)

Mais comum em varejo, loja pequena

Características:

- Impostos unificados em alíquota única
- Não calcula ICMS separado (está na alíquota única)
- Usa CSOSN 100, 101, 102, 103, 201, 202, 203
- Alíquota única varia por faixa de receita

Exemplo:

Alíquota única SN: 15% (exemplo)

Base: R\$ 100,00

Impostos: $100 \times 15 / 100 = R\$ 15,00$

Preço final: R\$ 100,00 (mantém o preço tabelado)

B. Lucro Presumido

Mais comum em empresas maiores

Características:

- ICMS = Alíquota ICMS por estado × Base
- PIS = Alíquota PIS por NCM × Base
- COFINS = Alíquota COFINS por NCM × Base
- IPI = Se aplicável (cartas geralmente isentas)

Exemplo:

Base: R\$ 100,00

ICMS (SP): 7% = R\$ 7,00

PIS (alíquota normal): 1,65% = R\$ 1,65

COFINS (alíquota normal): 7,6% = R\$ 7,60

IPI: 0% = R\$ 0,00

Total impostos: R\$ 16,25

Preço final: R\$ 100,00 (geralmente incluso)

4. Código para Cálculo

```
public class FiscalCalculoImpostos {

    /**
     * Calcula impostos para regime Simples Nacional
     */
    public static class ImpostoSimpleNacional {
        public double baseCalculo;
        public double icmsBase;          // Incluso na alíquota única
        public double pis;
        public double cofins;
        public double totalImposto;

        public ImpostoSimpleNacional calcular(double preco, double aliquotaSN) {
            this.baseCalculo = preco;
            // Em SN, tudo é uma alíquota única
            this.totalImposto = baseCalculo * aliquotaSN / 100;
            // Distribui proporcionalmente
            double proporcao = 15.0 / aliquotaSN; // 15% típico
            this.icmsBase = totalImposto * (7.0 / proporcao);
            this.pis = totalImposto * (1.25 / proporcao);
            this.cofins = totalImposto * (5.75 / proporcao);
            return this;
        }
    }

    /**
     * Calcula impostos para regime Lucro Presumido
     */
    public static class ImpostoLucroPresumido {
        public double baseCalculo;
        public double icms;
        public double pis;
        public double cofins;
        public double ipi;
        public double totalImposto;

        public ImpostoLucroPresumido calcular(
            double preco,
            double aliquotaICMS,
            double aliquotaPIS,
            double aliquotaCOFINS,
            double aliquotaIPI) {

            this.baseCalculo = preco;
            this.icms = baseCalculo * aliquotaICMS / 100;
            this.pis = baseCalculo * aliquotaPIS / 100;
            this.cofins = baseCalculo * aliquotaCOFINS / 100;
            this.ipi = baseCalculo * aliquotaIPI / 100;
            this.totalImposto = icms + pis + cofins + ipi;
            return this;
        }
    }
}
```

```

        }
    }
}
```

5. Alíquotas Padrão por NCM

NCM	Descrição	ICMS (SN)	ICMS (LP)	PIS	COFINS
95049090	Cartas colecionáveis	7%	7%	1,65%	7,6%
39229000	Sleeves/embalagens	7%	7%	1,65%	7,6%
59089000	Playmats	7%	7%	1,65%	7,6%

💡 Guia de Implementação Passo a Passo

FASE 1: Preparação (Semana 1)

Passo 1.1: Obter Certificado Digital

- [] Escolher fornecedor (Serasa, Certisign, Comodo)
- [] Solicitar Certificado A1 para NFCe
- [] Fazer download do arquivo .PFX
- [] Guardar senha com segurança
- [] Colocar arquivo em: data/certificados/certificado.pfx

Passo 1.2: Configurar Credenciais SEFAZ

- [] Acessar portal SEFAZ do seu estado
- [] Para NFCe: Gerar CSC e ID CSC
- [] Para NFe: Validar habilitação
- [] Testar conexão no ambiente de homologação

Passo 1.3: Criar Novas Tabelas SQL

```
// Em DB.java, adicionar no initSchema():

executeComLog(st,
    "CREATE TABLE IF NOT EXISTS imposto_icms (" +
    "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
    "estado TEXT NOT NULL, " +
    "estado_destino TEXT NOT NULL, " +
    "ncm TEXT NOT NULL, " +
    "aliquota_consumidor REAL, " +
    "aliquota_contribuinte REAL, " +
```

```

"ativo INTEGER DEFAULT 1, " +
"FOREIGN KEY(ncm) REFERENCES ncm(codigo), " +
"UNIQUE(estado, estado_destino, ncm) " +
")", "imposto_icms");

// ... (similar para IPI, PIS/COFINS, configuracao_nfe, fila_emissao, log)

```

Passo 1.4: Atualizar ConfigFiscalDAO

```

// Em ConfigFiscalDAO.java - adicionar métodos:
public void salvarConfiguracaoNFeNFCe(ConfiguracaoNFeNFCe config) throws
SQLException
public ConfiguracaoNFeNFCe obterConfiguracaoNFeNFCe() throws SQLException
public void atualizarAmbiente(String ambiente) throws SQLException

```

FASE 2: Infraestrutura de Comunicação (Semana 2)

Passo 2.1: Criar classe SEFAZWebServiceClient

```

// src/main/java/service/SEFAZWebServiceClient.java

package service;

import javax.xml.soap.*;
import java.io.*;
import java.net.*;

public class SEFAZWebServiceClient {

    private String urlWebservice;
    private String certificadoPath;
    private String certificadoSenha;
    private int timeout = 30000;

    public SEFAZWebServiceClient(String urlWebservice,
                                String certificadoPath,
                                String certificadoSenha) {
        this.urlWebservice = urlWebservice;
        this.certificadoPath = certificadoPath;
        this.certificadoSenha = certificadoSenha;
    }

    /**
     * Envia NF-e para autorização na SEFAZ
     * @param xmlAssinado XML já assinado digitalmente
     * @return RespostaAutorizacao com protocolo ou erro
     */
    public RespostaAutorizacao enviarNFe(String xmlAssinado) throws Exception {

```

```

        SOAPMessage request = criarRequestAutorizacao(xmlAssinado);
        SOAPMessage response = enviarSOAP(request);
        return extrairRespostaAutorizacao(response);
    }

    /**
     * Consulta status de NF-e autorizada
     */
    public RespostaConsulta consultarNFe(String chaveAcesso, String protocolo)
throws Exception {
    // Implementação similar
}

    /**
     * Cancela uma NF-e previamente autorizada
     */
    public RespostaCancelamento cancelarNFe(String chaveAcesso,
                                              String protocolo,
                                              String justificativa) throws Exception
{
    // Implementação
}

private SOAPMessage criarRequestAutorizacao(String xmlAssinado) throws
Exception {
    // Criar estrutura SOAP
    // Incluir certificado
    // Adicionar XML
    return soapMessage;
}

private SOAPMessage enviarSOAP(SOAPMessage request) throws Exception {
    // Configurar SSL/TLS com certificado
    // Fazer POST para URL
    // Retornar resposta
    return response;
}

private RespostaAutorizacao extrairRespostaAutorizacao(SOAPMessage response) {
    // Parse XML de resposta
    // Extrair protocolo e status
    // Retornar objeto
}
}

```

Passo 2.2: Criar classe FiscalXmlBuilder

```

// src/main/java/service/FiscalXmlBuilder.java

public class FiscalXmlBuilder {

```

```
/*
 * Constrói XML de NFCe conforme manual SEFAZ
 */
public static String construirXmlNFCe(DocumentoFiscalModel doc,
                                         ConfiguracaoNFeNFCe config,
                                         List<DocumentoFiscalItem> itens) throws
Exception {

    StringBuilder xml = new StringBuilder();
    xml.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
    xml.append("<NFe xmlns=\"http://www.portalfiscal.inf.br/nfe\">\n");

    // 1. Informe (que será assinado)
    xml.append("  <infNFe Id=\"NFe\"").append(doc.chaveAcesso).append("\"
versao=\"5.00\">\n");

    // 2. Identificação
    xml.append(construirIdentificacao(doc, config));

    // 3. Emitente
    xml.append(construirEmitente(config));

    // 4. Destinatário (consumidor)
    xml.append(construirDestinatario(doc));

    // 5. Itens
    for (DocumentoFiscalItem item : itens) {
        xml.append(construirItem(item, config));
    }

    // 6. Totais
    xml.append(construirTotais(doc, itens));

    // 7. Transporte (se houver)
    xml.append(construirTransporte());

    // 8. Cobranças
    xml.append(construirCobrancas(doc));

    // 9. Pag. (Pagamentos)
    xml.append(construirPagamentos(doc));

    // 10. Informações complementares
    xml.append(construirInfoCompl());

    xml.append("  </infNFe>\n");
    xml.append("</NFe>");

    return xml.toString();
}

private static String construirIdentificacao(DocumentoFiscalModel doc,
                                             ConfiguracaoNFeNFCe config) {
    StringBuilder sb = new StringBuilder();
    sb.append("  <id>NFe");
    sb.append(doc.chaveAcesso);
    sb.append("\n");
    sb.append("  <versao>5.00\n");
    sb.append("  <chaveAcesso>" + doc.chaveAcesso + "\n");
    sb.append("  <cnpj>00000000000000\n");
    sb.append("  <email>\n");
    sb.append("  <nome>\n");
    sb.append("  <identidade>\n");
    sb.append("  <transportador>\n");
    sb.append("  <destinatario>\n");
    sb.append("  <itens>\n");
    sb.append("  <totais>\n");
    sb.append("  <transporte>\n");
    sb.append("  <cobrancas>\n");
    sb.append("  <pagamentos>\n");
    sb.append("  <infoCompl>\n");
    return sb.toString();
}
```

```

        sb.append("      <ide>\n");
        sb.append("          <cUF>").append(ufParaCodigo(config.estado)).append("
</cUF>\n");
        sb.append("          <cNF>").append(generarCNF(doc.numero)).append("</cNF>\n");
        sb.append("          <assinaturaQRCode>").append(config.cscNfce).append("
</assinaturaQRCode>\n");
        sb.append("          <CNPJ>").append(config.cnpj).append("</CNPJ>\n");
        sb.append("          <modFe>65</modFe>\n"); // NFCe
        sb.append("          <serie>").append(config.serieNfce).append("</serie>\n");
        sb.append("          <nNF>").append(doc.numero).append("</nNF>\n");
        sb.append("          <dhEmi>").append(LocalDateTime.now().format(...)).append("</dhEmi>\n");
        sb.append("          <tpAmb>").append(config.ambiente.equals("producao") ? "2"
: "1").append("</tpAmb>\n");
        sb.append("          <tpEmis>1</tpEmis>\n"); // Normal
        sb.append("          <cDV>").append(cacularDV(doc.numero)).append("
</cDV>\n");
        sb.append("          <tpNF>1</tpNF>\n"); // Saída
        sb.append("          <idDest>1</idDest>\n"); // Operação interna
        sb.append("          <indFinal>1</indFinal>\n"); // Consumidor final
        sb.append("          <indPres>1</indPres>\n"); // Não se aplica
        sb.append("      </ide>\n");
    }

    // ... outros métodos (construirEmitente, construirDestinatario, etc.)
}

```

Passo 2.3: Criar classe FiscalDigitalSignature

```

// src/main/java/service/FiscalDigitalSignature.java

public class FiscalDigitalSignature {

    private KeyStore keyStore;
    private String certificadoPath;
    private String certificadoSenha;

    public FiscalDigitalSignature(String certificadoPath, String certificadoSenha)
        throws Exception {
        this.certificadoPath = certificadoPath;
        this.certificadoSenha = certificadoSenha;
        carregarCertificado();
    }

    /**
     * Carrega certificado .PFX em memória
     */
    private void carregarCertificado() throws Exception {
        keyStore = KeyStore.getInstance("PKCS12");
        FileInputStream fis = new FileInputStream(certificadoPath);

```

```
        keyStore.load(fis, certificadoSenha.toCharArray());
        fis.close();
    }

    /**
     * Assina XML usando certificado digital
     */
    public String assinarXml(String xmlDesassinado) throws Exception {
        // 1. Parse do XML
        DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document doc = builder.parse(new
ByteArrayInputStream(xmlDesassinado.getBytes()));

        // 2. Obter chave privada
        Enumeration<String> aliases = keyStore.aliases();
        String alias = aliases.nextElement();
        PrivateKey chavePrivada = (PrivateKey) keyStore.getKey(alias,
certificadoSenha.toCharArray());
        Certificate[] certificados = keyStore.getCertificateChain(alias);
        X509Certificate cert = (X509Certificate) certificados[0];

        // 3. Configurar XMLSigner
        XMLSignatureFactory signatureFactory =
XMLSignatureFactory.getInstance("DOM");
        Reference reference = signatureFactory.newReference(
            "#NFe" + gerarId(),

signatureFactory.newDigestMethod("http://www.w3.org/2001/04/xmlenc#sha256", null),
signatureFactory.newTransformMethod("http://www.w3.org/2000/09/xmldsig#enveloped-
signature", null),
            signatureFactory.newTransformMethod("http://www.w3.org/2001/10/xml-
exc-c14n#", null)
        );

        SignedInfo signedInfo = signatureFactory.newSignedInfo(
signatureFactory.newCanonicalizationMethod("http://www.w3.org/2001/10/xml-exc-
c14n#", null),
signatureFactory.newSignatureMethod("http://www.w3.org/2001/04/xmldsig-more#rsa-
sha256", null),
            Collections.singletonList(reference)
        );

        XMLSignature xmlSignature = signatureFactory.newXMLSignature(
            signedInfo,
            new X509KeyInfoProvider(cert, chavePrivada)
        );

        // 4. Assinar
        DOMSignContext signContext = new DOMSignContext(chavePrivada,
doc.getDocumentElement());
```

```

        xmlSignature.sign(signContext);

        // 5. Retornar XML assinado
        return xmlToString(doc);
    }

    private String xmlToString(Document doc) throws Exception {
        Transformer transformer =
TransformerFactory.newInstance().newTransformer();
        StringWriter stringWriter = new StringWriter();
        transformer.transform(new DOMSource(doc), new StreamResult(stringWriter));
        return stringWriter.toString();
    }
}

```

FASE 3: Integração com Vendas (Semana 3)

Passo 3.1: Atualizar DocumentoFiscalService

```

public class DocumentoFiscalService {

    /**
     * Criar e emitir NFCe ao finalizar venda
     */
    public void emitirNFCeParaVenda(int vendaId, String criadoPor) throws
Exception {
        try (Connection conn = DB.get()) {
            // 1. Obter dados da venda
            VendaModel venda = vendaDAO.obterPorId(vendaId);
            List<VendaItemModel> itens = vendaItemDAO.obterPorVenda(vendaId);
            ClienteModel cliente = clienteDAO.obterPorId(venda.clienteId);

            // 2. Criar documento fiscal pendente
            DocumentoFiscalModel documento =
criarDocumentoPendenteParaVenda(vendaId, criadoPor, "producao");

            // 3. Buscar configuração
            ConfiguracaoNFeNFCe config = configDAO.obterConfiguracao();
            if (!config.emitirNfce) {
                // Se desativado, apenas criar documento local
                return;
            }

            // 4. Construir XML
            List<DocumentoFiscalItem> docItens = converter(itens);
            String xmlDesassassinado = FiscalXmlBuilder.construirXmlNFCe(documento,
config, docItens);

            // 5. Assinar digitalmente
            FiscalDigitalSignature signer = new FiscalDigitalSignature(
                config.certificadoPathNfce,

```

```

        config.certificadoSenhaNfce
    );
    String xmlAssinado = signer.assinarXml(xmlDesassinado);

    // 6. Enviar para SEFAZ
    SEFAZWebServiceClient sefaz = new SEFAZWebServiceClient(
        config.urlWebserviceNfce,
        config.certificadoPathNfce,
        config.certificadoSenhaNfce
    );

    RespostaAutorizacao resposta = sefaz.enviarNFe(xmlAssinado);

    // 7. Atualizar documento com resposta
    if (resposta.autorizada()) {
        documento.status = "autorizada";
        documento.protocolo = resposta.protocolo;
        documento.chaveAcesso = resposta.chaveAcesso;
    } else {
        documento.status = "erro";
        documento.erro = resposta.mensagem;
    }

    documentoDAO.atualizar(documento);

    // 8. Gerar DANFE (nota fiscal visual)
    gerarDANFe(documento);

} catch (Exception e) {
    // Colocar na fila para retentativa
    filaEmissaoDAO.adicionar(vendaId, "NFCe", e.getMessage());
    throw e;
}
}

/**
 * Gerar DANFE (Documento Auxiliar da Nota Fiscal)
 */
private void gerarDANFe(DocumentoFiscalModel documento) throws Exception {
    // Usar biblioteca como DynamicReports ou iReport
    // Gerar PDF com QRCode
    // Salvar em data/danfe/
}
}

```

Passo 3.2: Atualizar VendaNovaDialog

```

// Em src/main/java/ui/venda/dialog/VendaNovaDialog.java

private void finalizarVenda() {
    // ... (código existente de finalização)
}

```

```

try {
    // Emitir NFCe se configurado
    DocumentoFiscalService docService = new DocumentoFiscalService();
    docService.emitirNFCeParaVenda(venda.getId(), usuario);

    JOptionPane.showMessageDialog(this,
        "Venda finalizada com sucesso!\n" +
        "NFCe autorizada com protocolo: " + documento.protocolo,
        "Sucesso",
        JOptionPane.INFORMATION_MESSAGE);

} catch (Exception e) {
    JOptionPane.showMessageDialog(this,
        "Venda salva, mas erro ao emitir NFCe:\n" + e.getMessage() +
        "\nTentaremos novamente em alguns minutos.",
        "Aviso",
        JOptionPane.WARNING_MESSAGE);
}
}

```

FASE 4: Fila de Processamento (Semana 4)

Passo 4.1: Criar classe FilaEmissaoFiscal

```

// src/main/java/service/FilaEmissaoFiscal.java

public class FilaEmissaoFiscal {

    private static final ScheduledExecutorService scheduler =
        Executors.newScheduledThreadPool(2);

    /**
     * Inicia processador de fila (deve ser chamado ao iniciar aplicação)
     */
    public static void iniciar() {
        scheduler.scheduleAtFixedRate(
            FilaEmissaoFiscal::processarFila,
            5, 5, TimeUnit.MINUTES // Processa a cada 5 min
        );
    }

    /**
     * Processa documentos pendentes da fila
     */
    private static void processarFila() {
        try {
            FilaEmissaoDAO dao = new FilaEmissaoDAO();
            List<FilaEmissaoModel> pendentes = dao.obterPendentes();

            for (FilaEmissaoModel item : pendentes) {

```

```

        if (item.tentativas >= item.maxTentativas) {
            dao.marcarFalha(item.id);
            continue;
        }

        try {
            DocumentoFiscalModel doc =
docDAO.obterPorId(item.documentoId);

            if ("NFCe".equals(item.tipo)) {
                processarNFCe(doc);
            } else if ("NFe".equals(item.tipo)) {
                processarNFe(doc);
            }

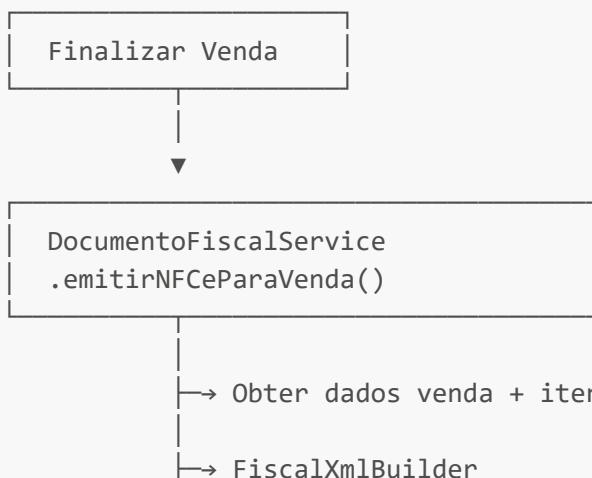
            dao.marcarSucesso(item.id);

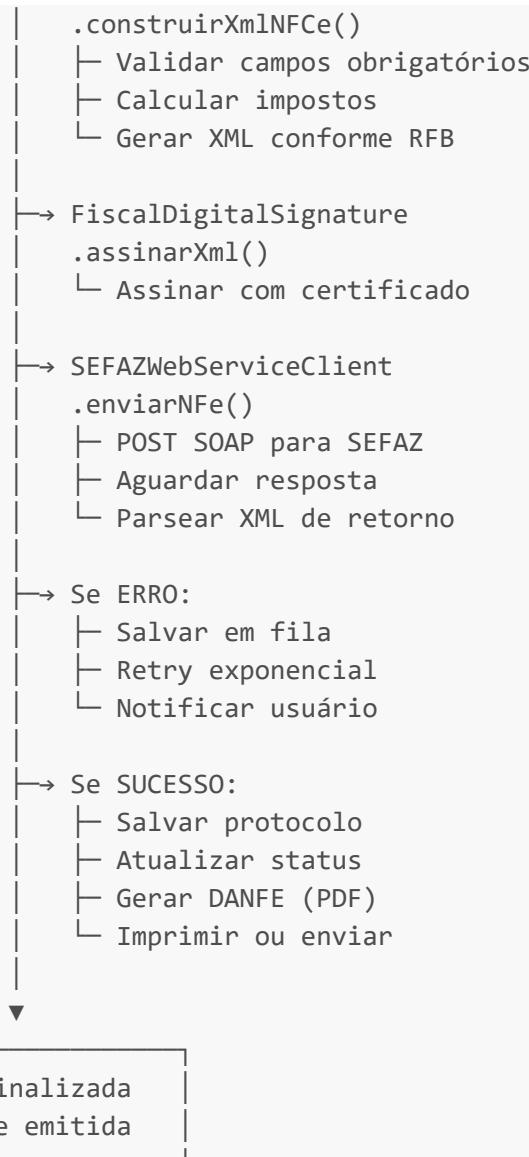
        } catch (Exception e) {
            int proximaTentativa = item.tentativas + 1;
            long delayMinutos = (long) Math.pow(2, proximaTentativa); // Backoff exponencial
            dao.marcarErro(item.id, e.getMessage(), delayMinutos);
        }
    } catch (Exception e) {
        e.printStackTrace(); // Log error
    }
}

private static void processarNFCe(DocumentoFiscalModel doc) throws Exception {
    // Lógica de envio/consulta similar ao passo 3.1
}
}

```

Fluxo de Emissão





⚠️ Tratamento de Erros e Retorno

Tipos de Erros Comuns

Erro	Código	Solução
Certificado expirado	-	Renovar certificado
CNPJ não habilitado	224	Validar habilitação na SEFAZ
Série/número inválido	573	Verificar sequência fiscal
CPF/CNPJ do cliente inválido	247	Validar documento do cliente
Alíquota ICMS não configurada	-	Adicionar NCM em imposto_icms
Timeout de conexão	-	Aumentar timeout ou usar contingência
Assinatura inválida	-	Verificar certificado e XML

Código de Tratamento

```
public class FiscalErrorHandler {

    public static class ErroFiscal extends Exception {
        public int codigo;
        public String mensagem;
        public boolean eh_retentavel;

        public ErroFiscal(int codigo, String msg, boolean retentavel) {
            super(msg);
            this.codigo = codigo;
            this.mensagem = msg;
            this.eh_retentavel = retentavel;
        }
    }

    public static ErroFiscal tratarErroSEFAZ(String respostaXml) throws ErroFiscal {
        // Parse XML de erro
        // Mapear código para mensagem
        // Decidir se é retentável

        if (contemErro(respostaXml)) {
            int codigo = extrairCodigoErro(respostaXml);
            String msg = extrairMensagemErro(respostaXml);
            boolean retentavel = codigoEhRetentavel(codigo);
            throw new ErroFiscal(codigo, msg, retentavel);
        }
    }

    private static boolean codigoEhRetentavel(int codigo) {
        // Erros de timeout, conexão = retentável
        // Erros de dados inválidos = não retentável
        return (codigo >= 500 && codigo <= 599) || // Erros de servidor
               codigo == 999; // Erro indefinido
    }
}
```

Testes e Validações

Checklist de Testes

- **Ambiente de Homologação**
 - Testar NFCe com dados fictícios
 - Validar XML gerado
 - Consultar autorização
 - Cancelar nota de teste
- **Certificado Digital**

- Verificar validade
- Testar assinatura
- Validar cadeia de certificados
- **Cálculo de Impostos**
 - Verificar ICMS por estado
 - Calcular PIS/COFINS
 - IPI (se aplicável)
- **Fila de Emissão**
 - Testar envio bem-sucedido
 - Simular erro de conexão
 - Verificar retentativa
- **DANFE**
 - Gerar PDF
 - QRCode válido
 - Impressão térmica

Dados de Teste Recomendados

```
// Testar com produto "Magic: The Gathering Booster"
VendaItemModel item = new VendaItemModel();
item.produtoId = "PRODUTO_001";
item.quantidade = 1;
item.preco = 50.00;
item.ncm = "95049090";
item.cfop = "5102";
item.csosn = "102"; // SN
item.origem = "0"; // Nacional

// Cliente consumidor
ClienteModel cliente = new ClienteModel();
cliente.nome = "Consumidor";
cliente.cpf = "00000000000"; // Genérico ou real para testes
```

💻 Dependências Maven Necessárias

```
<!-- Assinatura digital -->
<dependency>
    <groupId>org.apache.santuario</groupId>
    <artifactId>xmлsec</artifactId>
    <version>2.3.1</version>
</dependency>
```

```
<!-- XML parsing/building -->
<dependency>
    <groupId>javax.xml</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.3.1</version>
</dependency>
<dependency>
    <groupId>com.sun.xml.bind</groupId>
    <artifactId>jaxb-impl</artifactId>
    <version>2.3.1</version>
</dependency>

<!-- SOAP -->
<dependency>
    <groupId>javax.xml.soap</groupId>
    <artifactId>javax.xml.soap-api</artifactId>
    <version>1.4.0</version>
</dependency>
<dependency>
    <groupId>com.sun.xml.messaging.saaj</groupId>
    <artifactId>saaj-impl</artifactId>
    <version>1.5.3</version>
</dependency>

<!-- HTTP Client -->
<dependency>
    <groupId>org.apache.httpcomponents.client5</groupId>
    <artifactId>httpclient5</artifactId>
    <version>5.2.1</version>
</dependency>

<!-- PDF para DANFE -->
<dependency>
    <groupId>com.itextpdf</groupId>
    <artifactId>itextpdf</artifactId>
    <version>5.5.13.3</version>
</dependency>

<!-- QR Code -->
<dependency>
    <groupId>com.google.zxing</groupId>
    <artifactId>core</artifactId>
    <version>3.5.1</version>
</dependency>
<dependency>
    <groupId>com.google.zxing</groupId>
    <artifactId>javase</artifactId>
    <version>3.5.1</version>
</dependency>
```

Proteção de Dados Sensíveis

```
// NUNCA: Armazenar senha em texto plano
String senhaPlana = "12345"; // X ERRADO

// SIM: Criptografar senha armazenada
String senhaCriptografada = AESUtils.encrypt(senha, chaveSecreta);
configDAO.salvar("certificado_senha_nfce", senhaCriptografada);

// Ao usar:
String senhaDescriptografada = AESUtils.decrypt(senhaCriptografada, chaveSecreta);
```

Validação de Entrada

```
// Sempre validar antes de enviar para SEFAZ
public void validarXmlAntesDeSend(String xml) throws ErroFiscal {
    if (xml == null || xml.isBlank()) {
        throw new ErroFiscal(001, "XML vazio", false);
    }

    // Validar contra XSD da RFB
    XMLValidator.validarContraXSD(xml, "nfe_v5.00.xsd");

    // Verificar campos obrigatórios
    if (!itemCampo(xml, "infNFe")) {
        throw new ErroFiscal(002, "Falta infNFe", false);
    }
}
```

📞 Contatos e Recursos

- **SEFAZ Nacional:** <https://www.sefaz.fazenda.gov.br/>
- **Portal NFC-e:** <https://www.nfce.fazenda.gov.br/>
- **Manual RFB NFe 5.00:** <https://www.gov.br/infraestrutura/pt-br/assuntos/tecnologia-da-informacao/nfe>
- **Certificadoras:** Serasa, Certisign, Comodo
- **Suporte:** contato@sefaz.seu_estado.gov.br

Documento finalizado - Pronto para implementação!