GERENCIAMENTO DE CONFIGURAÇÃO DE SOFTWARE – AVALIAÇÃO N2

Disciplina: Gerência de Configuração de Software

Sistema: Gestão de Estoque e Caixa

Formato: Documentação técnica (PDF)

Aluno: Daniel Douglas Melo Gomes Neto e José Carlos Vaz

Data: 17/11/2025

1 - ITENS DE CONFIGURAÇÃO E ESTRUTURA DO PROJETO

1.1. Itens de Configuração (ICs)

Backend (Java Spring Boot)

- **Controllers** (Camada REST)
    - AuthController.java: Gerencia login, validação de token e logout.
    - ProdutoController.java: CRUD de produtos, movimentações de estoque e histórico.
    - UsuarioController.java: CRUD de usuários e gestão de perfis.
    - VendaController.java: Registro de vendas e consultas (por ID, por usuário, listagem).
- **Services** (Regras de Negócio)
    - AuthService.java: Lógica de autenticação e geração de tokens de sessão.
    - MovimentacaoEstoqueService.java: Regras para entrada, saída e ajuste de inventário (audit trail).
    - ProdutoService.java: Validações de produtos e controle de saldo.
    - UsuarioService.java: Regras de unicidade de e-mail e criação de usuários.
    - VendaService.java: Lógica de baixa automática de estoque, cálculo de troco e totalização da venda.
- **Entities** (Modelo de Dados)
    - Modelos: Usuario.java, Produto.java, Venda.java, VendaItem.java, MovimentacaoEstoque.java.
    - Enums*:* PerfilUsuario.java, StatusUsuario.java, TipoMovimentacao.java.
- **Repositórios** (Acesso a Dados)
    - UsuarioRepository.java, ProdutoRepository.java, VendaRepository.java, MovimentacaoEstoqueRepository.java.

- **Arquivos de Configuração e DTOs**
    - application.properties: Configurações do H2 Database, porta do servidor (8080) e Swagger.
    - OpenAPIConfig.java: Configuração da documentação Swagger/OpenAPI.
    - *DTOs Principais:* LoginRequestDTO, VendaRequestDTO, ProdutoRequestDTO, EstoqueMovimentacaoRequestDTO.

Frontend (Angular)

- **Componentes** (Telas)
    - LoginComponent: Tela de autenticação.
    - MainLayoutComponent: Estrutura base (Sidebar e Topbar).
    - DashboardComponent: Visão geral para administradores (KPIs).
    - WelcomeComponent: Tela inicial do operador.
    - UsuariosComponent: Gestão de usuários (CRUD).
    - EstoqueComponent: Gestão de produtos e histórico de movimentações.
    - CaixaComponent: PDV (Ponto de Venda).
    - RelatoriosComponent: Listagem e filtros de vendas.
- **Services** (Comunicação API)
    - AuthService.ts, ProdutoService.ts, UsuarioService.ts, VendaService.ts.
- **Rotas e Guardas**
    - app.routes.ts: Definição de navegação.
    - auth.guard.ts: Protege rotas contra acesso não autenticado.
    - role.guard.ts: Controla acesso baseado no perfil (ADMIN/OPERADOR).

Banco de Dados

- **H2 Database (Em memória)**: Utilizado para desenvolvimento ágil.
- **Tabelas Principais**: usuarios, produtos, vendas, venda_itens, movimentacoes_estoque.
- **Scrip SQL de criação:**

```
-- Tabela de Usuários

CREATE TABLE IF NOT EXISTS usuarios (

    id BIGINT AUTO_INCREMENT PRIMARY KEY,

    nome_completo VARCHAR(255) NOT NULL,
```

```
    email VARCHAR(255) UNIQUE NOT NULL,
    senha VARCHAR(255) NOT NULL,
    perfil VARCHAR(10) NOT NULL CHECK (perfil IN ('ADMIN',
'OPERADOR')),
    status VARCHAR(10) NOT NULL CHECK (status IN ('ATIVO',
'INATIVO'))
);

-- Tabela de Produtos
CREATE TABLE IF NOT EXISTS produtos (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    codigo VARCHAR(255) UNIQUE NOT NULL,
    nome VARCHAR(255) NOT NULL,
    categoria VARCHAR(255) NOT NULL,
    quantidade_estoque DECIMAL(10,3) NOT NULL,
    preco_unitario DECIMAL(10,2) NOT NULL
);

-- Tabela de Vendas
CREATE TABLE IF NOT EXISTS vendas (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    data_hora TIMESTAMP NOT NULL,
    valor_total DECIMAL(10,2) NOT NULL,
    valor_recebido DECIMAL(10,2) NOT NULL,
    troco DECIMAL(10,2) NOT NULL,
    usuario_id BIGINT NOT NULL,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id)
);
```

```
-- Tabela de Itens de Venda
CREATE TABLE IF NOT EXISTS venda_itens (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    venda_id BIGINT NOT NULL,
    produto_id BIGINT NOT NULL,
    quantidade DECIMAL(10,3) NOT NULL,
    preco_unitario DECIMAL(10,2) NOT NULL,
    subtotal DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (venda_id) REFERENCES vendas(id),
    FOREIGN KEY (produto_id) REFERENCES produtos(id)
);

-- Tabela de Movimentações de Estoque
CREATE TABLE IF NOT EXISTS movimentacoes_estoque (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    data_hora TIMESTAMP NOT NULL,
    tipo VARCHAR(1) NOT NULL CHECK (tipo IN ('E', 'S', 'I')),
    produto_id BIGINT NOT NULL,
    quantidade DECIMAL(10,3) NOT NULL,
    estoque_anterior DECIMAL(10,3) NOT NULL,
    estoque_atual DECIMAL(10,3) NOT NULL,
    motivo VARCHAR(100) NOT NULL,
    usuario_id BIGINT,
    FOREIGN KEY (produto_id) REFERENCES produtos(id),
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id)
);
```

- **Documentos: Documento de arquitetura geral da aplicação**
- **Arquivos de configuração: aplication.properties (backend), OpenAPIConfig (Backend), package.json (Frontend)**

1.2. Estrutura do Projeto (Conceitual)

Estrutura do repositório do projeto:

```
syspdv-project/
├── backend/
│   ├── src/main/java/com/fatesg/syspdv_backend/
│   │   ├── controller/       # Pontos de entrada da API (REST)
│   │   ├── service/          # Lógica de negócio e transações
│   │   ├── repository/       # Interfaces JPA para o banco de dados
│   │   ├── model/            # Entidades JPA (Tabelas)
│   │   ├── dto/              # Objetos de transferência de dados
(Input/Output)
│   │   └── config/           # Configurações globais (Swagger, CORS)
│   └── src/main/resources/
│       └── application.properties # Configuração do ambiente
├── frontend/
│   ├── src/app/
│   │   ├── core/             # Recursos globais (Services, Models, Guards)
│   │   ├── layout/           # Componentes estruturais (Menu, Header)
│   │   └── pages/            # Telas funcionais (Login, Caixa, Estoque)
│   └── src/environments/     # Variáveis de ambiente
```

Descrição da Organização:

A separação clara entre Frontend e Backend permite o desenvolvimento independente das camadas (Separação da equipe). No Backend, a arquitetura em camadas (MVC/Service Layer) garante que as regras de negócio (como a baixa de estoque ao vender) fiquem isoladas dos controladores. No Frontend, a divisão entre core (lógica reutilizável) e pages (visualização) facilita a manutenção e a escalabilidade do sistema.

2 - BASELINES E VERSIONAMENTO

2.1. Baseline BL0 - Inicial

- **Data:** 03/11/2025
- **Descrição:** Criação inicial do projeto Backend e Frontend
- **Conteúdo da BL0:**
    - Backend: Projeto Spring Boot criado com dependências (Web, JPA, H2, Validation) e conexão com banco de dados testada.
    - Frontend: Projeto Angular gerado com PrimeNG instalado e configuração de roteamento básico.
- **Critério de Estabilidade:** O backend inicia sem erros na porta 8080 e o frontend compila e exibe a página inicial na porta 4200.

2.2. Baseline BL1 - Funcional

- **Data:** 04/11/2025
- **Descrição:** Versão funcional com os módulos essenciais operando (MVP).
- **Requisitos Entregues:**
    - Autenticação (Login/Logout) com controle de sessão.
    - Gestão de Usuários (CRUD) e Estoque (CRUD + Movimentações).
    - Ponto de Venda (Caixa) registrando vendas e baixando estoque.
    - Relatórios operacionais.
- **Critério de Estabilidade:** Todos os fluxos principais (Login -> Venda -> Relatório) foram testados e não apresentam erros. O histórico de movimentação de estoque está sendo gravado corretamente.

2.3. Política de Versionamento

Estratégia Adotada: Semantic Versioning (SemVer)

Regras de Versão:

- **MAJOR (X.0.0):** Mudanças profundas na arquitetura (ex: Migração de H2 para PostgreSQL ou mudança de framework Frontend).
- **MINOR (0.X.0):** Novas funcionalidades (ex: Adicionar módulo de "Contas a Pagar" ou nova forma de pagamento).
- **PATCH (0.0.X):** Correções de bugs (ex: Ajuste na validação de senha ou correção visual no CSS).

Tabela de Exemplos:

Versão	Descrição	Tipo

v0.1.0	Estrutura inicial (BL0)	Inicial
v0.5.0	Módulos de Cadastro (Usuário/Produto) prontos	MINOR
v1.0.0	Sistema completo com Vendas e Relatórios (BL1)	MAJOR
v1.0.1	Correção no alinhamento dos botões do Estoque	PATCH

Relação Baselines → Versionamento:

- BL0 → v0.1.0
- BL1 → v1.0.0

3 - RASTREABILIDADE + RFC

3.1. Matriz de Rastreabilidade

ID	Requisito	Arquivo(s) Relacionado(s)	Observação
RF-01	Autenticação de Usuário	AuthController.java, AuthService.java, LoginComponent.ts, auth.guard.ts	Login valida senha e gera token de sessão armazenado no LocalStorage.
RF-02	Movimentação de Estoque (Histórico)	ProdutoController.java, MovimentacaoEstoqueService.java, EstoqueComponent.ts	Implementado endpoint /historico e tela com abas para entrada/saída.

RF-03	Registro de Venda (PDV)	VendaController.java, VendaService.java, CaixaComponent.ts, Venda.java	Ao finalizar venda, o VendaService chama o método de baixa de estoque automaticamente.
RF-04	Validação de Estoque Negativo	ProdutoService.java (método baixarEstoque), CaixaComponent.ts	Validação dupla: Frontend bloqueia adicionar ao carrinho e Backend lança exceção se saldo < 0.

3.2. RFC - Solicitação de Mudança (Proposta)

Como o campo "Categoria" já existe no seu projeto, elaborei uma RFC para uma melhoria de segurança/negócio muito comum e que não está no código atual: ***Forma de Pagamento****.*

- **Código da RFC:** RFC-002
- 1. Descrição da Mudança:
  Adicionar o atributo "Forma de Pagamento" na Venda (ex: Dinheiro, Pix, Cartão), pois atualmente o sistema assume apenas um valor monetário genérico.
- 2. Justificativa:
  Permitir relatórios financeiros mais detalhados, separando o que é dinheiro em caixa do que entrará via operadora de cartão.
- **3. Impacto nos Itens de Configuração:**

Item	Tipo de Alteração
Venda.java (Entity)	Adicionar campo private FormaPagamento formaPagamento;
FormaPagamento.java (Enum)	Criar Enum com valores: DINHEIRO, CARTAO, PIX
VendaRequestDTO.java	Adicionar campo para receber a forma de pagamento do front
CaixaComponent.html	Adicionar um p-dropdown para selecionar a forma de pagamento antes de finalizar

- **4. Riscos:**
  - Vendas antigas ficarão com esse campo nulo no banco de dados.

    - **Mitigação:** Definir DINHEIRO como valor padrão (default) para registros legados na migração.

- **5. Status:** APROVADA
- **6. Versão de Implementação:** v1.1.0
- **7. Data:** 18/11/2024