

Relatório de Especificação do Software (TypeScript)

1. Visão Geral

Nome do Projeto: Sistema de Gerenciamento para Farmácia

Tecnologia: TypeScript

Arquitetura: API REST + Frontend separada

Objetivo: Construir um sistema que automatize os processos de estoque, vendas e relatórios de uma farmácia, baseada em uma especificação clara e detalhada que poderá servir de base para geração de código.

Metodologia: Spec-Driven Development (SDD)

Este documento será a fonte de verdade para todas as fases do desenvolvimento.

2. Objetivos do Sistema

- Gerenciar estoque de produtos (entrada, saída, validade e categorias).
- Registrar vendas, incluindo produtos controlados.
- Gerar relatórios de estoque e financeiro.
- Garantir rastreabilidade e consistência de dados desde o cadastro até os relatórios.
- Permitir escalabilidade e manutenção eficiente no futuro.

3. Requisitos do Sistema

Requisitos Funcionais (RF)

| Código | Descrição |
|--------|---|
| RF01 | Registrar produtos com dados completos (nome, validade, quantidade, preço). |
| RF02 | Atualizar estoque automaticamente após uma venda. |
| RF03 | Emitir alertas de produtos próximos da validade. |
| RF04 | Registrar vendas e seus itens associados. |
| RF05 | Gerar relatórios de estoque e financeiro. |

Requisitos Não Funcionais (RNF)

| Código | Descrição |
|--------|---|
| RNF01 | O sistema deve responder a requisições em até 2 segundos sob carga normal. |
| RNF02 | O código deve ser modular e legível, baseado em TypeScript. |
| RNF03 | Deve suportar testes automatizados desde a primeira versão. |
| RNF04 | Dados sensíveis devem ser validados e sanitizados antes de qualquer operação. |

4. Especificação Arquitetural

4.1 Estrutura Geral

O projeto será dividido em módulos logicamente separados:

- Módulo de Estoque: entidades, serviços e controladores relacionados aos produtos.
- Módulo de Vendas: lógica de registro de vendas e cálculo de totais.
- Relatórios: geração de relatórios financeiros e de inventário.
- Infraestrutura: configuração do servidor, rotas e banco de dados.

Essa separação segue princípios de Domain-Driven Design (DDD) para manter o domínio de negócio bem isolado.

5. Modelagem do Domínio

Entidade: Produto

```
ts

interface Produto {
  id: string;
  nome: string;
  categoria: string;
  quantidadeEstoque: number;
  validade: Date;
  preco: number;
  controlado: boolean;
}
```

Entidade: Venda

```
ts

interface Venda {
  id: string;
  data: Date;
  itens: ItemVenda[];
  total: number;
  formaPagamento: "dinheiro" | "cartão";
}
```

ItemVenda

```
ts

interface ItemVenda {
  produtoId: string;
  quantidade: number;
  precoUnitario: number;
}
```

6. Endpoints da API

| Rota | Método | Descrição |
|------------------------|--------|----------------------------|
| /produtos | GET | Listar produtos |
| /produtos | POST | Criar novo produto |
| /vendas | POST | Registrar uma venda |
| /relatorios/estoque | GET | Gerar relatório de estoque |
| /relatorios/financeiro | GET | Gerar relatório financeiro |

7. Workflow de Desenvolvimento (SDD)

Etapa 1 — Especificação Inicial

Documentar todos os requisitos de comportamento, dados e restrições antes de escrever qualquer código.

Etapa 2 — Validação da Spec

Revisão da documentação com stakeholders para garantir que os requisitos estão corretos.

Etapa 3 — Planejamento

Converter a spec em planos técnicos detalhados (ex.: estruturas de dados, contratos de API e formatos JSON).

Etapa 4 — Implementação

Gerar código de acordo com a spec, podendo inclusive integrar com ferramentas que geram código a partir da documentação.

Etapa 5 — Testes

Criar testes automatizados que confirmem que o software segue a especificação.

8. Critérios de Aceitação

- Cada funcionalidade deve estar coberta por testes;
- O sistema deve refletir fielmente as specs definidas;
- Relatórios devem ser exportáveis em formatos úteis (CSV/PDF).

9. Benefícios do SDD nesse projeto

- Redução de ambiguidade entre requisitos e código;
- Facilita uso de IA para geração de partes do sistema;
- Permite ter um artefato executável como fonte de verdade;
- Reduz retrabalho por falta de documentação clara;
- Ao seguir Spec-Driven Development, sua equipe garante mais clareza e organização, preparando o projeto para manutenção e evolução com menos erros.