Documentação Técnica

API REST para Gerenciamento de Vendas

Data da Avaliação: 31/03/2025

Desenvolvedor: Cairo Maranzatto

1. Visão Geral do Projeto

1.1 Objetivo

Este documento apresenta a implementação de uma API REST para gerenciamento de vendas, desenvolvida como parte da avaliação técnica para a vaga de Desenvolvedor Sênior. O projeto demonstra a aplicação de conceitos avançados de desenvolvimento, arquitetura de software e boas práticas de programação.

1.2 Escopo

- Desenvolvimento de API REST para gestão de vendas
- Implementação de regras de negócio complexas
- Sistema de descontos baseado em quantidade
- Gestão de eventos de domínio
- Documentação completa

> **Nota:** Esta solução foi desenvolvida seguindo os princípios de Clean Architecture e Domain-Driven Design, garantindo alta coesão, baixo acoplamento e facilidade de manutenção.

2. Arquitetura e Estrutura do Projeto

2.1 Decisões Arquiteturais

Arquitetura Base

- Clean Architecture
- DDD (Domain-Driven Design)
- CQRS (Command Query Responsibility Segregation)
- Princípios SOLID

Padrões Adicionais

- Repository Pattern
- Unit of Work
- Specification Pattern
- Event Sourcing

2.2 Estrutura do Projeto

```
Ambev.DeveloperEvaluation/
├── src/
│   ├── WebApi/          # Camada de API/Interface
│   ├── Application/     # Camada de Aplicação
│   ├── Domain/          # Camada de Domínio
│   ├── ORM/            # Camada de Persistência
│   ├── IoC/            # Injeção de Dependência
│   └── Common/         # Componentes Compartilhados
└── tests/              # Testes Unitários e de Integração
```

2.3 Fluxo de Dados

1. **API Layer** - Recebe requisições HTTP, valida entrada, mapeia DTOs e chama Application Layer
2. **Application Layer** - Processa comandos/queries, coordena operações, publica eventos e chama Domain Layer

3. **Domain Layer** - Contém regras de negócio, valida entidades, publica eventos e define interfaces
4. **ORM Layer** - Implementa persistência, mapeia entidades, gerencia transações e otimiza queries

2.4 Dependências Principais

- .NET 7.0
- Entity Framework Core
- SQL Server
- AutoMapper
- FluentValidation
- Swagger/OpenAPI
- Docker
- xUnit (testes)

3. Mapeamento Detalhado do Projeto

3.1 Camada de Domínio (Domain)

Categoria	Componentes	Descrição
Entidades	Sale.cs, SaleItem.cs, Product.cs, Customer.cs, Branch.cs	Entidades principais do domínio com comportamento e regras de negócio
Eventos	SaleCreatedEvent.cs, SaleCancelledEvent.cs, ItemCancelledEvent.cs	Eventos de domínio para comunicação entre agregados
Repositórios	ISaleRepository.cs, ISaleItemRepository.cs	Interfaces para persistência das entidades
Validações	SaleValidation.cs, SaleItemValidation.cs	Regras de validação complexas do domínio
Serviços	ISaleService.cs, SaleService.cs	Serviços de domínio para operações complexas

3.2 Camada de Aplicação (Application)

Módulo	Componentes	Descrição
Vendas	CreateSaleCommand, CancelSaleCommand, GetSaleQuery, SaleEventHandler	Casos de uso relacionados a vendas
Usuários	CreateUserCommand, GetUserQuery	Gerenciamento de usuários
Autenticação	LoginCommand, TokenResponse	Autenticação JWT

3.3 Camada de Persistência (ORM)

Categoria	Componentes	Descrição
Contexto	DefaultContext.cs	DbContext do Entity Framework
Mapeamentos	SaleMapping.cs, SaleItemMapping.cs, ProductMapping.cs	Configurações de mapeamento ORM
Repositórios	SaleRepository.cs, SaleItemRepository.cs	Implementações concretas dos repositórios

3.4 Camada de API (WebApi)

Categoria	Componentes	Descrição
Controladores	SalesController.cs, UsersController.cs, AuthController.cs	Endpoints REST da API
Middleware	ErrorHandlingMiddleware.cs, AuthenticationMiddleware.cs	Tratamento global de erros e autenticação
Configuração	Program.cs, appsettings.json, Dockerfile	Configuração da aplicação e containerização

3.5 Testes

- **Testes Unitários:** Domínio, aplicação e repositório
- **Testes de Integração:** API e banco de dados

4. Funcionalidades Implementadas

4.1 Gestão de Vendas

- CRUD completo de vendas
- Gestão de itens de venda
- Cálculo automático de descontos
- Sistema de cancelamento

4.2 Regras de Negócio

Exemplo de implementação do cálculo de descontos:

```
public void CalculateDiscount() { if (Quantity < 4) { Discount = 0; return; } if (Quantity >= 10 && Quantity <= 20) { Discount = 0.20m; return; } if (Quantity >= 4) { Discount = 0.10m; return; } Discount = 0; }
```

4.3 Eventos de Domínio

- SaleCreatedEvent - Disparado quando uma nova venda é criada
- SaleCancelledEvent - Disparado quando uma venda é cancelada
- ItemCancelledEvent - Disparado quando um item de venda é cancelado
- Sistema completo de logging de eventos

4.4 Documentação da API

- Documentação automática via Swagger/OpenAPI
- README em inglês e português
- Mapeamento completo do projeto

5. Aspectos Técnicos Destacados

5.1 Segurança

- Autenticação JWT
- Validação de entrada robusta
- Proteção contra injeção SQL
- Logging seguro de informações sensíveis

5.2 Performance

- Queries otimizadas com EF Core
- Cache implementado para dados estáticos
- Operações assíncronas em toda a stack
- Carregamento eficiente de relacionamentos

5.3 Escalabilidade

- Arquitetura preparada para escala horizontal
- Containerização com Docker
- Health checks para monitoramento
- Logging estruturado para análise

5.4 Qualidade e Manutenibilidade

- Código limpo e organizado
- Nomes descritivos e significativos
- Comentários relevantes apenas onde necessário
- Cobertura abrangente de testes
- Separação clara de responsabilidades

6. Conclusão

A implementação desta API de gerenciamento de vendas demonstra a aplicação de conceitos avançados de arquitetura de software e boas práticas de desenvolvimento. A solução foi projetada para ser robusta, escalável e de fácil manutenção, atendendo aos requisitos técnicos e funcionais especificados.

A arquitetura baseada em Clean Architecture e DDD permite que o sistema evolua de forma sustentável, com regras de negócio claramente definidas e isoladas da infraestrutura. A implementação de CQRS e Event Sourcing proporciona flexibilidade para futuras extensões do sistema.