AgroERP Enterprise

Desenvolvimento Full Stack de um Sistema de Gestão com Arquitetura Corporativa Segura

Contexto

MVP simulando **ambiente corporativo real**

Objetivo

Sistema ERP para agronegócio focado em **gestão de estoque e vendas**

Diferencial

Arquitetura escalável, segurança robusta e **RBAC**

Stack Tecnológico

Backend

- Java 21 LTS
- Spring Boot 3.5
- Injeção de dependências
- API REST

Frontend

- React.js
- Vite Hot-reload
- SPASingle Page Application
- Componentização

Database

- MySQL 8.0
- Hibernate/JPA
- Mapeamento ORM
- Consultas otimizadas

Segurança

- Spring Security
- JWTJSON Web Token
- BCrypt Hash de senhas
- Autenticação stateless

UI/UX

- Material UI Design System
- RechartsBusiness Intelligence
- Componentes reutilizáveis
- Design responsivo

Documentação

- OpenAPI
- Swagger API Docs
- Documentação automática
- Interface interativa

Arquitetura de Software

Padrão MVC em Camadas

Controller

Gerencia requisições HTTP

Service

Regras de negócio complexas

Repository

Acesso direto aos dados

Padrões de Segurança

DTO (Data Transfer Object) — Desacoplamento

Garante segurança e filtragem de dados sensíveis

@Transactional — Atomicidade

Garante integridade entre baixa de estoque e registro de venda

Fluxo de Requisição

1. Requisição HTTP chega ao **Controller**
2. Controller invoca o **Service** com dados validados
3. Service executa regras de negócio e chama o **Repository**
4. Repository realiza operações no banco de dados
5. Resposta com **DTO** retorna através das camadas

Implementação de Segurança

Autenticação Stateless

Uso de **Tokens JWT** para validar sessões sem sobrecarregar o servidor

- Tokens autocontidos com informações de usuário
- Sem necessidade de armazenar estado no servidor

Criptografia

Senhas armazenadas com **Hash BCrypt** nunca em texto puro

- Algoritmo de hashing adaptativo com salt
- Proteção contra ataques de rainbow table

Filtro de Segurança

SecurityFilter customizado para interceptar requisições e validar o token antes de chegar ao Controller

- Validação de assinatura e expiração do token
- Injeção do contexto de segurança no request

Fluxo de Autenticação e CORS

Login
Credenciais enviadas via POST

Geração do Token
JWT criado com informações do usuário

Validação
SecurityFilter verifica o token

CORS
Comunicação segura entre React (porta 5173) e Java (porta 8080)

Acesso
Request autorizado com contexto de segurança

Regras de Negócio e RBAC

Role-Based Access Control

Sistema de **permissões hierárquicas** que define o que cada tipo de usuário pode visualizar e fazer no sistema

ADMIN

- Dashboard Financeiro Completo
- Visualização de todas as vendas
- Permissão para deletar registros

VENDEDOR

- Acesso restrito ao Estoque
- Apenas vendas próprias
- Bloqueio de dados sensíveis

Solução Técnica

1. Service verifica **contexto de segurança**
2. Identifica **perfil** do usuário (ADMIN/VENDEDOR)
3. Altera **query** do banco de dados
4. **findAll** vs **findByVendedorId**

```
if (securityContext.hasRole("ADMIN")) {
  return repository.findAll();
} else {
  return repository.findByVendedorId(
    securityContext.getCurrentUserId());
}
```

Frontend e Experiência do Usuário

Dashboard Interativo

Gráficos em tempo real usando Recharts

Gráfico de Pizza

Visualização de distribuição de vendas por categoria

Curva ABC

Análise de produtos mais vendidos e rentáveis

Responsividade e Layout

Layout adaptável via Grid System do Material UI

Desktop

Experiência otimizada para telas maiores

Mobile

Interface adaptada para dispositivos móveis

Experiência do Usuário

Feedback Visual

Substituição de alerts nativos por Snackbars profissionais

```
const handleSuccess = () => {
  enqueueSnackbar('Operação realizada com sucesso!', {
    variant: 'success'
  });
};
```

Data Formatting

Tratamento de moeda (BRL) e datas no padrão brasileiro

```
const formatCurrency = (value) => {
  return new Intl.NumberFormat('pt-BR', {
    style: 'currency',
    currency: 'BRL'
  }).format(value);
};
```

Persistência e Massa de Dados

Data Seeder Automatizado

Script **CommandLineRunner** complexo para popular o sistema com dados realistas

1. **Criação de Usuários Padrão**
   Admin e Vendedor com perfis e permissões distintas

2. **População de Produtos**
   500 produtos com preços aleatórios e categorias variadas

3. **Simulação de Vendas**
   Centenas de transações distribuídas entre os usuários

```
// Exemplo de implementação do Data Seeder
@Component
public class DataSeeder implements CommandLineRunner {
  @Override
  public void run(String... args) throws Exception {
    createDefaultUsers();
    populateProducts(500);
    simulateSales(200);
  }
}
```

Benefícios do Data Seeder

Teste de Performance

Avaliação imediata do sistema com volume realista de dados

Teste de Paginação

Validação de queries com grandes conjuntos de dados

Demonstração Rápida

Sistema funcional imediatamente após o deploy

Dados para Dashboard

Visualização de gráficos com dados reais desde o início

Modelo de Banco de Dados

Estrutura de Tabelas

usuarios

id PK | nome VARCHAR | login VARCHAR | perfil ENUM | senha_hash VARCHAR

produtos

id PK | nome VARCHAR | part_number VARCHAR | estoque INT | preco DECIMAL

vendas

id PK | data_hora DATETIME | produto_id FK | quantidade INT | valor_total DECIMAL | vendedor_id FK

Otimizações de Performance

- Índices em chaves estrangeiras
- Índice em part_number
- Índice em data_hora
- Índice composto em vendedor_id + data_hora

Relacionamentos e Integridade

Relacionamento Vendas ↔ Produtos

Cada venda está associada a um produto através de **produto_id** (chave estrangeira)

Relacionamento Vendas ↔ Usuários

Cada venda é registrada por um vendedor através de **vendedor_id** (chave estrangeira)

Segregação de Dados por Perfil

Consultas filtradas por **perfil** (ADMIN/VENDEDOR) no backend

Transacionalidade de Vendas

@Transactional garante consistência entre baixa de estoque e registro de venda

Resultados e Conclusão

Produto Final

- Sistema **Full Stack** funcional
- Segurança robusta com **RBAC**
- Documentação completa via Swagger

Aprendizados

- Domínio do ciclo de vida do Spring
- Tratamento de erros de CORS
- Fluxo de autenticação JWT
- Integração Front-Back eficiente

100%	500+	2
Funcionalidade	Produtos	Perfis de Acesso

Próximos Passos (Roadmap)

Containerização com Docker

Empacotamento da aplicação para ambientes de produção

Testes Unitários JUnit/Mockito

Garantir estabilidade e qualidade do código

Deploy em Cloud AWS/Render

Disponibilização do sistema em ambiente de produção