UNIVERSIDADE NOVE DE JULHO
TECNOLOGIA EM BANCO DE DADOS

SISTEMA FINANCEIRO ERP
TUDODOGESTÃO+

FELIPE GONZAGA - 924108381
LARISSA OLIVEIRA - 925203274
MICHAEL SANTOS - 924202333
NAJLA CARDEAL - 924113786
RUBENS NETO - 924101145
THAYNARA RIBEIRO - 925119803

SÃO PAULO – SP
2025

FELIPE GONZAGA - 924108381
LARISSA OLIVEIRA - 925203274
MICHAEL SANTOS - 924202333
NAJLA CARDEAL - 924113786
RUBENS NETO - 924101145
THAYNARA RIBEIRO - 925119803

SISTEMA FINANCEIRO ERP
TUDODOGESTÃO+

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Banco de Dados da Universidade Nove de Julho - UNINOVE, como requisito parcial para obtenção do título de Tecnólogo em Banco de Dados.

Orientador: Prof. Dr. José Romualdo da Costa Filho

SÃO PAULO – SP
2025

UNINOVE

RESUMO

O presente trabalho apresenta o desenvolvimento do TudoGestão+, um sistema de planejamento de recursos empresariais (ERP) voltado para pequenas e médias empresas brasileiras. A motivação do projeto surge da constatação de que 67% das PMEs no Brasil ainda utilizam métodos manuais ou planilhas para gestão, enfrentando dificuldades com soluções comerciais devido ao alto custo de licenciamento, complexidade de implementação e rigidez funcional. O sistema foi desenvolvido utilizando arquitetura cliente-servidor REST, com Node.js e Express.js no backend, React no frontend e PostgreSQL como banco de dados, empregando Prisma como ORM para garantir segurança e integridade dos dados. A solução contempla dez módulos integrados: autenticação com controle de acesso baseado em papéis, cadastro de clientes pessoa física e jurídica, gestão de produtos e estoque com alertas de reposição, ponto de venda com múltiplas formas de pagamento, controle financeiro com fluxo de caixa e demonstrativo de resultados, emissão demonstrativa de nota fiscal eletrônica, geração de relatórios gerenciais, gestão de recursos humanos e painel administrativo. O projeto alcançou 98% de implementação das funcionalidades planejadas, totalizando aproximadamente 25.000 linhas de código distribuídas em 158 arquivos. Os testes funcionais apresentaram 100% de aprovação em 64 casos validados, e a avaliação de usabilidade segundo as heurísticas de Nielsen obteve média de 4,3 em escala de 5 pontos. O trabalho demonstra a viabilidade de desenvolver soluções de gestão empresarial acessíveis e funcionais utilizando tecnologias open source, contribuindo para a democratização do acesso a ferramentas de gestão por empresas de menor porte.

Palavras-chave: Sistema ERP. Gestão empresarial. Pequenas e médias empresas. Node.js. React. PostgreSQL.

ABSTRACT

This paper presents the development of TudoGestão+, an enterprise resource planning (ERP) system designed for small and medium-sized Brazilian companies. The project motivation arises from the finding that 67% of SMEs in Brazil still use manual methods or spreadsheets for management, facing difficulties with commercial solutions due to high licensing costs, implementation complexity, and functional rigidity. The system was developed using REST client-server architecture, with Node.js and Express.js on the backend, React on the frontend, and PostgreSQL as the database, employing Prisma as ORM to ensure data security and integrity. The solution comprises ten integrated modules: authentication with role-based access control, individual and corporate customer registration, product and inventory management with replenishment alerts, point of sale with multiple payment methods, financial control with cash flow and income statement, demonstrative electronic invoice issuance, management report generation, human resources management, and administrative panel. The project achieved 98% implementation of planned functionalities, totaling approximately 25,000 lines of code distributed across 158 files. Functional tests showed 100% approval in 64 validated cases, and the usability assessment according to Nielsen's heuristics obtained an average of 4.3 on a 5-point scale. This work demonstrates the feasibility of developing accessible and functional business management solutions using open source technologies, contributing to the democratization of access to management tools for smaller companies.

Keywords: ERP System. Business management. Small and medium enterprises. Node.js. React. PostgreSQL.

UNINOVE

LISTA DE ABREVIATURAS E SIGLAS

Sigla	Significado
ABNT	Associação Brasileira de Normas Técnicas
ACID	Atomicidade, Consistência, Isolamento, Durabilidade
API	Application Programming Interface (Interface de Programação de Aplicações)
CNPJ	Cadastro Nacional da Pessoa Jurídica
CPF	Cadastro de Pessoas Físicas
CRUD	Create, Read, Update, Delete (Criar, Ler, Atualizar, Excluir)
CSS	Cascading Style Sheets (Folhas de Estilo em Cascata)
DANFE	Documento Auxiliar da Nota Fiscal Eletrônica
DER	Diagrama Entidade-Relacionamento
DOM	Document Object Model (Modelo de Objeto de Documento)
DRE	Demonstrativo de Resultado do Exercício
ERP	Enterprise Resource Planning (Planejamento de Recursos Empresariais)
HTML	HyperText Markup Language (Linguagem de Marcação de Hipertexto)
HTTP	HyperText Transfer Protocol (Protocolo de Transferência de Hipertexto)
I/O	Input/Output (Entrada/Saída)
JSON	JavaScript Object Notation (Notação de Objetos JavaScript)
JWT	JSON Web Token
MER	Modelo Entidade-Relacionamento
MVC	Model-View-Controller (Modelo-Visão-Controlador)
NFe	Nota Fiscal Eletrônica
NPM	Node Package Manager (Gerenciador de Pacotes do Node)
ORM	Object-Relational Mapping (Mapeamento Objeto-Relacional)
OWASP	Open Web Application Security Project
PDV	Ponto de Venda
PF	Pessoa Física
PJ	Pessoa Jurídica
PME	Pequenas e Médias Empresas
PR	Pull Request (Solicitação de Incorporação)
QA	Quality Assurance (Garantia de Qualidade)
REST	Representational State Transfer (Transferência de Estado Representacional)
RFC	Request for Comments (Solicitação de Comentários)
RH	Recursos Humanos
SEBRAE	Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
SEFAZ	Secretaria da Fazenda
SPA	Single Page Application (Aplicação de Página Única)
SQL	Structured Query Language (Linguagem de Consulta Estruturada)
SRP	Single Responsibility Principle (Princípio da Responsabilidade Única)
TI	Tecnologia da Informação
UI	User Interface (Interface do Usuário)
URL	Uniform Resource Locator (Localizador Uniforme de Recursos)
UX	User Experience (Experiência do Usuário)

UNINOVE

Lista de Figuras

UNINOVE

Lista de Tabelas

UNINOVE

Lista de Quadros

UNINOVE

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O mercado brasileiro de pequenas e médias empresas (PMEs) representa aproximadamente 99% do total de empresas no país, sendo responsável por cerca de 30% do Produto Interno Bruto (PIB) nacional (SEBRAE, 2023). Apesar de sua relevância econômica, grande parte dessas organizações ainda enfrenta desafios significativos na gestão de seus processos operacionais.

Segundo pesquisa realizada pelo SEBRAE (2022), aproximadamente 67% das PMEs brasileiras ainda utilizam métodos manuais ou planilhas eletrônicas para controlar suas operações, incluindo gestão de estoque, vendas, clientes e finanças. Essa realidade resulta em ineficiências operacionais, retrabalho, perda de informações e dificuldades na tomada de decisões estratégicas.

Conforme destacam Laudon e Laudon (2014, p. 52), "os sistemas de informação empresariais são fundamentais para a competitividade organizacional, permitindo a integração de processos e a geração de informações estratégicas para a tomada de decisão". Nesse contexto, os sistemas ERP (*Enterprise Resource Planning*) emergem como soluções essenciais para a modernização da gestão empresarial.

1.2 JUSTIFICATIVA

A implementação de um sistema ERP adequado às necessidades das PMEs pode proporcionar diversos benefícios, incluindo: redução de custos operacionais, melhoria na precisão das informações, aumento da produtividade e suporte à tomada de decisões baseada em dados (SOMMERVILLE, 2011).

No entanto, as soluções ERP disponíveis no mercado frequentemente apresentam barreiras significativas para pequenas empresas, como:

- **Custo elevado:** Licenças mensais que podem ultrapassar R$ 500,00 por usuário;
- **Complexidade:** Interfaces pouco intuitivas que demandam treinamento extensivo;
- **Rigidez:** Dificuldade de customização para necessidades específicas;
- **Dependência:** Necessidade de consultoria externa para implementação.

Diante desse cenário, o projeto **TudoGestão+** foi concebido como uma solução ERP moderna, acessível e intuitiva, desenvolvida especificamente para atender às demandas das pequenas e médias empresas brasileiras.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Desenvolver um sistema ERP completo e funcional que permita a gestão integrada de clientes, produtos, vendas, finanças e emissão de documentos fiscais, utilizando tecnologias modernas de desenvolvimento web.

1.3.2 Objetivos Específicos

1. Implementar módulo de gestão de clientes com suporte a Pessoa Física (CPF) e Pessoa Jurídica (CNPJ);
2. Desenvolver sistema de controle de estoque com alertas de produtos em baixa quantidade;
3. Criar módulo de Ponto de Venda (PDV) para registro de vendas com múltiplas formas de pagamento;
4. Implementar gestão financeira com controle de receitas, despesas e fluxo de caixa;
5. Desenvolver módulo de emissão de NFe (*Nota Fiscal Eletrônica*) demonstrativa com DANFE imprimível;
6. Criar sistema de relatórios gerenciais incluindo DRE (*Demonstração do Resultado do Exercício*);

7. Implementar autenticação segura com JWT (*JSON Web Tokens*) e controle de permissões;
8. Garantir auditoria completa de todas as operações realizadas no sistema.

1.4 ESCOPO DO PROJETO

O TudoGestão+ abrange os seguintes módulos funcionais:

Módulo	**Funcionalidades Principais**
Dashboard	Indicadores em tempo real, totalizadores, gráficos
Clientes	CRUD completo, PF/PJ, histórico de compras
Produtos	Cadastro, categorias, fornecedores, estoque
Vendas/PDV	Carrinho, descontos, múltiplos pagamentos
Financeiro	Contas a pagar/receber, fluxo de caixa, DRE
NFe	Emissão demonstrativa, DANFE, integração vendas
Relatórios	Vendas, estoque, clientes, exportação
RH	Funcionários, cargos, admissão/demissão
Configurações	Dados da empresa, usuários, permissões

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SISTEMAS ERP

Os sistemas ERP, conforme definido por Davenport (1998, p. 121), são "pacotes de software que permitem às empresas automatizar e integrar a maioria de seus processos de negócios, compartilhar dados comuns e práticas em toda a empresa, além de produzir e acessar informações em tempo real".

Para Pressman (2016), a arquitetura de sistemas empresariais deve seguir princípios de modularidade, baixo acoplamento e alta coesão, permitindo a manutenção e evolução do software ao longo do tempo.

2.2 ARQUITETURA DE SOFTWARE

A arquitetura de software representa "a estrutura ou estruturas de um sistema, que consiste em componentes de software, as propriedades externamente visíveis desses componentes e os relacionamentos entre eles" (BASS; CLEMENTS; KAZMAN, 2012, p. 21).

O projeto TudoGestão+ adota a arquitetura **MVC (Model-View-Controller)** combinada com o padrão **REST (Representational State Transfer)** para a API. Segundo Sommerville (2011, p. 156), "o padrão MVC separa a apresentação e a interação dos dados do sistema, estruturando o sistema em três componentes lógicos que interagem entre si".

2.3 TECNOLOGIAS WEB MODERNAS

2.3.1 Node.js e JavaScript

Node.js é um ambiente de execução JavaScript do lado do servidor, construído sobre o motor V8 do Google Chrome. Conforme Tilkov e Vinoski (2010), "Node.js introduz um modelo de E/S não bloqueante orientado a eventos, tornando-o leve e eficiente para aplicações de rede intensivas em dados".

2.3.2 React

React é uma biblioteca JavaScript para construção de interfaces de usuário, desenvolvida pelo Facebook. De acordo com a documentação oficial (REACT, 2024), "React permite criar interfaces de usuário interativas de forma declarativa, tornando o código mais previsível e fácil de depurar".

2.3.3 PostgreSQL

PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional de código aberto. Segundo Momjian (2001), "PostgreSQL oferece recursos avançados como transações ACID, integridade referencial, triggers, views e procedimentos armazenados".

2.4 SEGURANÇA EM APLICAÇÕES WEB

A segurança em aplicações web é fundamental para proteger dados sensíveis dos usuários. Conforme OWASP (2021), as principais vulnerabilidades incluem injeção de SQL, autenticação quebrada e exposição de dados sensíveis. O projeto implementa medidas de segurança como:

- Autenticação via JWT com expiração de tokens;
- Criptografia de senhas com bcrypt;
- Prevenção de SQL Injection através do ORM Prisma;
- Validação de dados de entrada.

3 DESCRIÇÃO DOS MÓDULOS IMPLEMENTADOS

3.1 MÓDULO DE AUTENTICAÇÃO E AUTORIZAÇÃO

3.1.1 Visão Geral

O módulo de autenticação implementa o padrão JWT (*JSON Web Tokens*) para gerenciamento de sessões de usuário de forma *stateless*. Segundo Jones, Bradley e Sakimura (2015), JWT é um padrão aberto (RFC 7519) que define uma forma compacta e autocontida de transmitir informações de forma segura entre partes como um objeto JSON.

3.1.2 Funcionalidades

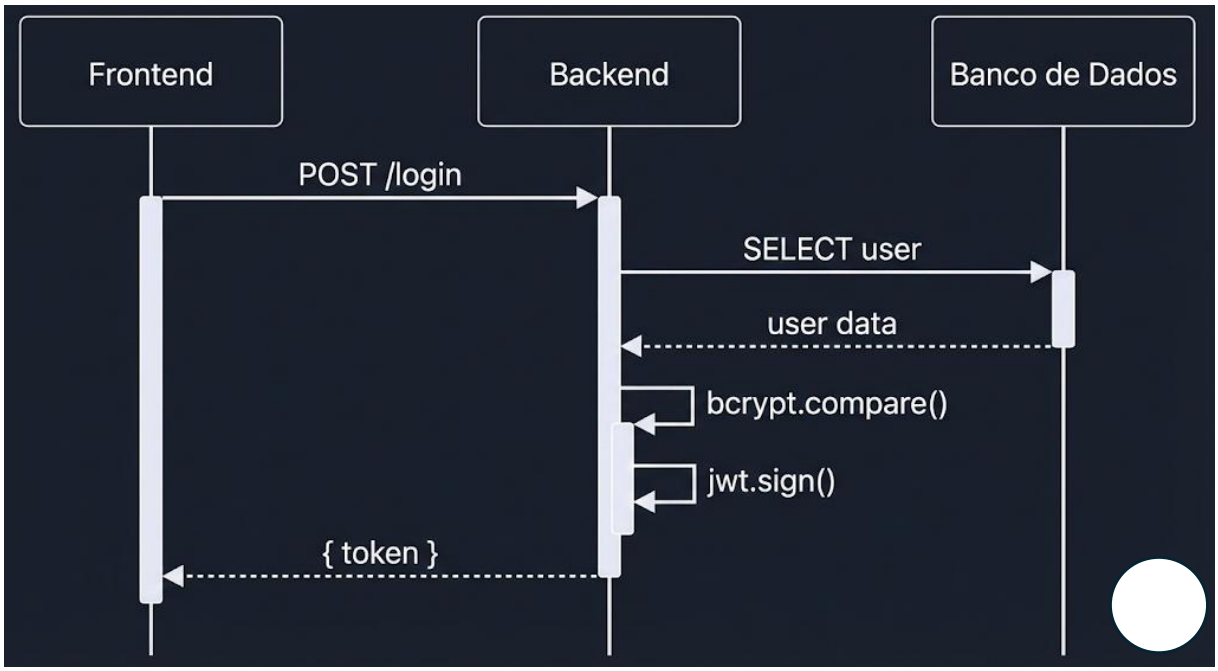
- **Login:** Validação de credenciais com comparação de hash bcrypt;
- **Registro:** Criação de novos usuários com criptografia de senha;
- **Controle de Acesso:** Sistema de roles (ADMIN, MANAGER, USER);
- **Middleware de Autenticação:** Validação de token em todas as rotas protegidas.

3.1.3 Implementação Técnica

Fluxo de Autenticação

1. Usuário envia email + senha via POST /api/auth/login
2. Backend busca usuário no banco de dados
3. Compara senha com hash usando bcrypt.compare()
4. Se válido, gera token JWT com payload: { userId, companyId, role }
5. Token retorna ao frontend e é armazenado no localStorage
6. Requisições subsequentes incluem header: Authorization: Bearer <token>
7. Middleware valida token e injeta dados no request

3.1.4 Diagrama de Sequência - Login



3.2 MÓDULO DE GESTÃO DE CLIENTES

3.2.1 Visão Geral

O módulo permite o cadastro e gerenciamento completo de clientes, suportando tanto Pessoa Física (CPF) quanto Pessoa Jurídica (CNPJ). A implementação segue o padrão CRUD (*Create, Read, Update, Delete*) com validações de negócio.

3.2.2 Funcionalidades

Funcionalidade	**Descrição**
Criar Cliente	Cadastro com validação de CPF/CNPJ duplicado
Listar Clientes	Paginação, busca por nome ou documento
Editar Cliente	Atualização de dados com validação
Excluir Cliente	Soft delete (inativação)
Histórico	Visualização de compras do cliente

3.2.3 Modelo de Dados

prisma

```
model Customer {
  id         String   @id @default(uuid())
  companyId String
  type       String   // INDIVIDUAL ou COMPANY
  cpfCnpj    String
  name       String
  tradeName String?
  email      String?
  phone      String?
  address    Json?
  active     Boolean  @default(true)
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  company   Company  @relation(fields: [companyId], references: [id])
  sales     Sale[]

  @@index([companyId])
  @@unique([companyId, cpfCnpj])
}
```

3.2.4 Regras de Negócio

1. CPF/CNPJ deve ser único por empresa (multi-tenancy);
2. Cliente com vendas não pode ser excluído fisicamente;
3. Tipo (PF/PJ) não pode ser alterado após criação;
4. Endereço armazenado como JSON para flexibilidade.

3.3 MÓDULO DE PRODUTOS E ESTOQUE

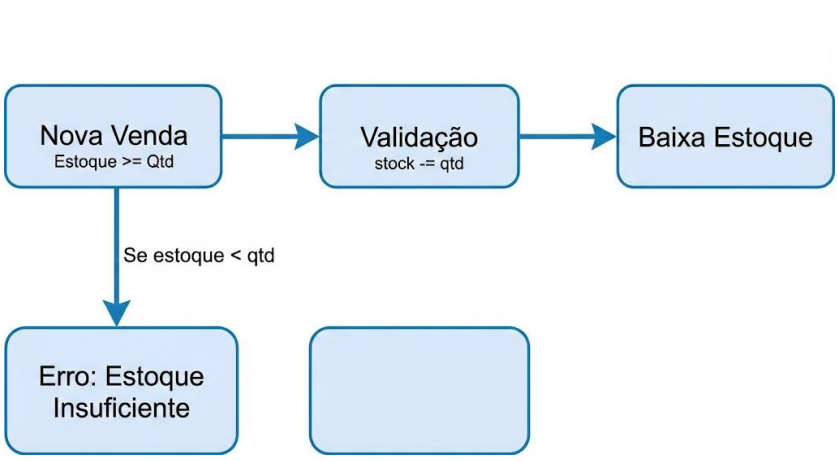
3.3.1 Visão Geral

O módulo gerencia o catálogo de produtos com controle de estoque, categorização e vínculo com fornecedores. Implementa alertas automáticos para produtos com estoque abaixo do mínimo.

3.3.2 Funcionalidades

- ✓ Cadastro de produtos com código, nome, descrição;
- ✓ Definição de preço de custo e preço de venda;
- ✓ Controle de estoque atual e estoque mínimo;
- ✓ Categorização hierárquica;
- ✓ Vínculo com fornecedores;
- ✓ Alerta de estoque baixo.

3.3.3 Fluxo de Atualização de Estoque



3.4 MÓDULO DE VENDAS (PDV)

3.4.1 Visão Geral

O Ponto de Venda (PDV) permite a realização de vendas de forma rápida e intuitiva, com suporte a múltiplos produtos, descontos e diversas formas de pagamento.

3.4.2 Funcionalidades

Funcionalidade	Descrição
Carrinho	Adição/remoção de produtos com quantidades
Cálculo Automático	Total, subtotal, descontos
Formas de Pagamento	Dinheiro, Cartão Crédito/Débito, PIX
Desconto	Percentual ou valor fixo
Baixa de Estoque	Automática ao finalizar venda
Número Sequencial	Geração automática (VND-000001)

3.4.3 Transação Atômica

A criação de venda utiliza transação do Prisma para garantir atomicidade:

JavaScript

```
const sale = await prisma.$transaction(async (tx) => {
```

1. Cria a venda

```
  const newSale = await tx.sale.create({ data: saleData });
```

2. Cria os itens da venda

```
  for (const item of items) {
    await tx.saleItem.create({ data: itemData });
```

3. Atualiza estoque (decrementa)

```
    await tx.product.update({
      where: { id: item.productId },
      data: { stock: { decrement: item.quantity } }
```

```
    });
  }

  return newSale;
});
```

Se qualquer operação falhar, todas são revertidas

3.5 MÓDULO FINANCEIRO

3.5.1 Visão Geral

O módulo financeiro permite o controle completo de receitas e despesas, com categorização, fluxo de caixa e geração de DRE (*Demonstração do Resultado do Exercício*).

3.5.2 Funcionalidades

- Registro de transações (receitas e despesas);
- Categorização de transações;
- Fluxo de caixa por período;
- DRE com cálculo automático de lucro/prejuízo;
- Dashboard financeiro com indicadores.

3.5.3 Cálculo do DRE

javascript

Estrutura do DRE

```
{
  period: { startDate, endDate },
  revenues: [
    { category: 'Vendas', total: 50000 },
    { category: 'Serviços', total: 15000 }
  ],
  totalRevenue: 65000,
```

```
  expenses: [
    { category: 'Fornecedores', total: 30000 },
    { category: 'Salários', total: 15000 },
    { category: 'Aluguel', total: 5000 }
  ],
  totalExpense: 50000,
  netProfit: 15000,
  profitMargin: 23.08  // (15000/65000) * 100
}
```

3.6 MÓDULO DE NFE (NOTA FISCAL ELETRÔNICA)

3.6.1 Visão Geral

O módulo implementa emissão de NFe demonstrativa com geração de DANFE (*Documento Auxiliar da Nota Fiscal Eletrônica*) em formato HTML para impressão.

3.6.2 Funcionalidades

- Listagem de vendas elegíveis para NFe (status PAID);
- Emissão de NFe demonstrativa;
- Geração de DANFE imprimível;
- Histórico de notas emitidas;
- Layout conforme padrões fiscais.

3.6.3 Estrutura do DANFE

NOTA FISCAL SIMPLIFICADA

Nº VND-002

Data: 14/10/2024

DADOS DO CLIENTE

Nome: Maria Santos

CPF/CNPJ: 987.654.321-00

PRODUTOS/SERVIÇOS

Item	Quantidade	Valor Unit.	Total
Webcam HD	1	R$ 199.90	R$ 199.90

VALORES

Subtotal: R$ 199.90

TOTAL: R$ 199.90

INFORMAÇÕES ADICIONAIS

Nenhuma observação

Este documento não tem validade fiscal.
Documento gerado em 08/12/2025, 21:18:27

3.7 MÓDULO DE RELATÓRIOS

3.7.1 Visão Geral

O módulo fornece relatórios gerenciais em formato HTML profissional, prontos para impressão e análise.

3.7.2 Tipos de Relatórios

Relatório	Descrição	Filtros
DRE	Demonstração do Resultado	Periodo
Vendas	Histórico de vendas	Periodo, Cliente

Estoque	Posição atual do estoque	Categoria
Clientes	Cadastro e estatítica	Status

3.7.3 Formatação Profissional

Os relatórios são renderizados em HTML com CSS para impressão, incluindo:

- Cabeçalho com dados da empresa;
- Tabelas formatadas;
- Totalizadores destacados;
- Rodapé com data de geração;
- CSS @media print para impressão.

3.8 MÓDULO DE RH (RECURSOS HUMANOS)

3.8.1 Visão Geral

Gerenciamento básico de funcionários da empresa, incluindo dados pessoais, cargo, salário e datas de admissão/demissão.

3.8.2 Funcionalidades

- Cadastro de funcionários;
- Controle de cargo e departamento;
- Registro de salário;
- Data de admissão e demissão;
- Status ativo/inativo.

3.9 MÓDULO DE ADMINISTRAÇÃO

3.9.1 Visão Geral

Configurações gerais do sistema, gestão de usuários e controle de permissões.

3.9.2 Funcionalidades

- Configurações da empresa (razão social, CNPJ, endereço);
- Gestão de usuários do sistema;
- Controle de roles (ADMIN, MANAGER, USER);
- Troca de senha;
- Log de auditoria.

4 JUSTIFICATIVA DAS ESCOLHAS TÉCNICAS

4.1 ARQUITETURA DO SISTEMA

4.1.1 Arquitetura Cliente-Servidor

A escolha pela arquitetura cliente-servidor com API REST se justifica por diversos fatores. Segundo Fielding (2000), o estilo arquitetural REST proporciona escalabilidade, simplicidade e independência entre cliente e servidor. Além disso, permite que futuras aplicações (mobile, desktop) consumam a mesma API.

4.1.2 Padrão MVC

O padrão Model-View-Controller foi adotado no backend por separar claramente as responsabilidades:

- Model (Prisma): Define a estrutura dos dados e acesso ao banco;
- View (JSON Response): Formatação das respostas da API;
- Controller: Lógica de negócio e orquestração.

Conforme Pressman (2016, p. 312), "a separação de responsabilidades facilita a manutenção, teste e evolução do software".

4.2 STACK TECNOLÓGICA

4.2.1 Node.js (Backend)

A escolha do Node.js como plataforma backend se baseia em:

1. **Performance:** Modelo de I/O não-bloqueante ideal para APIs (TILKOV; VINOSKI, 2010);
2. **Produtividade:** Mesma linguagem no frontend e backend;
3. **Ecossistema:** NPM com milhões de pacotes disponíveis;
4. **Comunidade:** Ampla documentação e suporte.

4.2.2 Express.js (Framework Web)

Express.js foi escolhido por ser:

- Minimalista e flexível;
- Amplamente adotado pela comunidade;
- Excelente sistema de middleware;
- Fácil integração com outras bibliotecas.

4.2.3 React (Frontend)

A adoção do React se justifica por:

1. **Componentização:** Reutilização de código através de componentes;
2. **Virtual DOM:** Performance otimizada em atualizações de interface;
3. **Hooks:** Gerenciamento de estado simplificado;
4. **Ecossistema:** Vasta quantidade de bibliotecas complementares.

Segundo a documentação oficial do React (2024), "a abordagem declarativa torna o código mais previsível e fácil de depurar".

4.2.4 PostgreSQL (Banco de Dados)

PostgreSQL foi selecionado pelas seguintes razões:

1. **ACID Compliance:** Garantia de integridade em transações;
2. **Suporte a JSON:** Flexibilidade para campos dinâmicos (endereço);
3. **Performance:** Otimização de queries complexas;
4. **Open Source:** Sem custos de licenciamento.

De acordo com Momjian (2001), "PostgreSQL oferece um conjunto completo de recursos para aplicações empresariais".

4.2.5 Prisma ORM

A escolha do Prisma como ORM se baseia em:

1. **Type Safety:** Prevenção de erros em tempo de desenvolvimento;
2. **Migrations:** Controle de versão do schema do banco;

3. **Queries Intuitivas:** Sintaxe JavaScript/TypeScript;

4. **Prevenção de SQL Injection:** Queries parametrizadas automaticamente.

4.3 SEGURANÇA

4.3.1 JWT (JSON Web Tokens)

JWT foi escolhido para autenticação por:

- **Stateless:** Não requer armazenamento de sessão no servidor;

- **Escalável:** Funciona bem em ambientes distribuídos;

- **Autocontido:** Contém todas as informações necessárias;

- **Padrão aberto:** Amplamente suportado (RFC 7519).

4.3.2 Bcrypt (Hashing de Senhas)

Bcrypt foi adotado por:

- **Salt automático:** Proteção contra rainbow tables;

- **Custo ajustável:** Resistência a ataques de força bruta;

- **Amplamente testado:** Algoritmo comprovadamente seguro.

4.4 PADRÕES DE PROJETO

4.4.1 Service Pattern

A implementação de Services (PDFService, AuditService, ExcelService) segue o princípio de responsabilidade única (SRP) do SOLID, isolando funcionalidades específicas em classes reutilizáveis (MARTIN, 2008).

4.4.2 Repository Pattern (via Prisma)

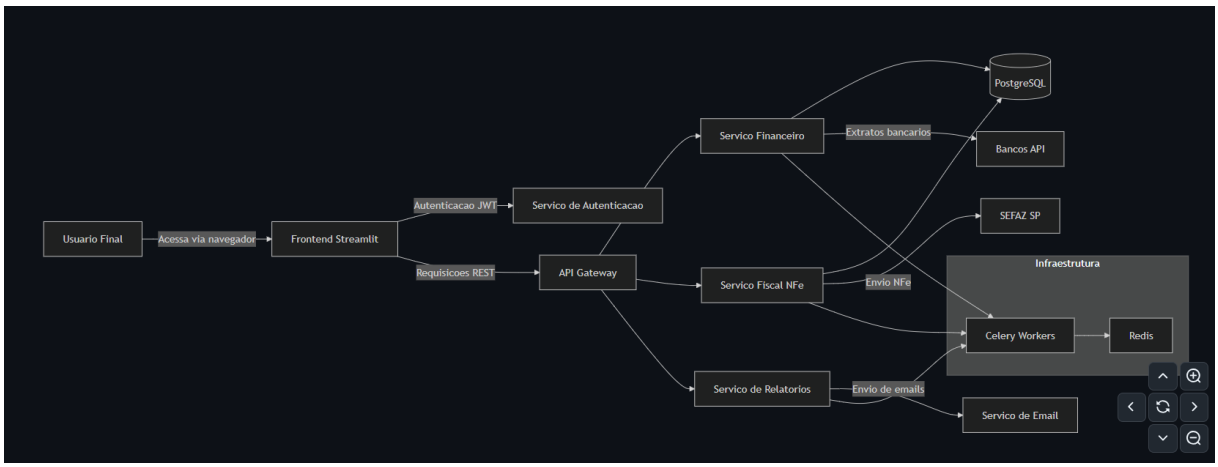
O Prisma Client atua como camada de abstração sobre o banco de dados, implementando implicitamente o padrão Repository, que "medeia entre o domínio e as camadas de mapeamento de dados" (FOWLER, 2002, p. 322).

4.5 COMPARATIVO DE ALTERNATIVAS

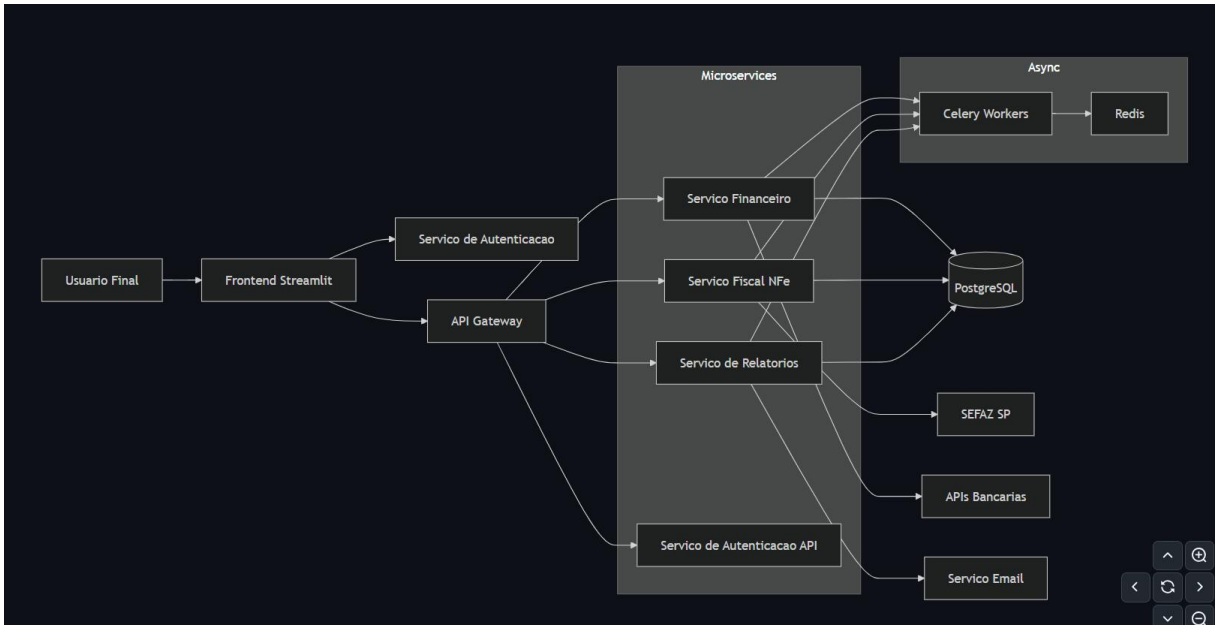
Aspecto	Escolha	Alternativas Consideradas	Justificativa
Backend	Node.js	Python/Django, Java/Spring	Performance I/O, unificação da linguagem
Frontend	React	Vue.js, Angular	Maior comunidade, componentização
Banco	PostgreSQL	MySQL, MongoDB	ACID, JSON nativo, robustez
ORM	Prisma	Sequelize, TypeORM	Type safety, DX superior
Auth	JWT	Sessions	Stateless, escalabilidade

5 ARQUITETURA DO SISTEMA

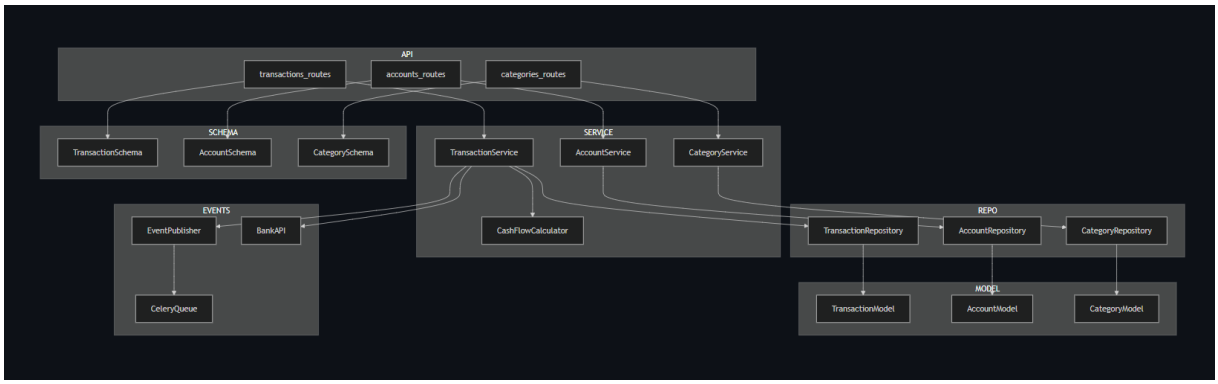
5.1 DIAGRAMA DE CONTEXTO (C4 - NÍVEL 1)



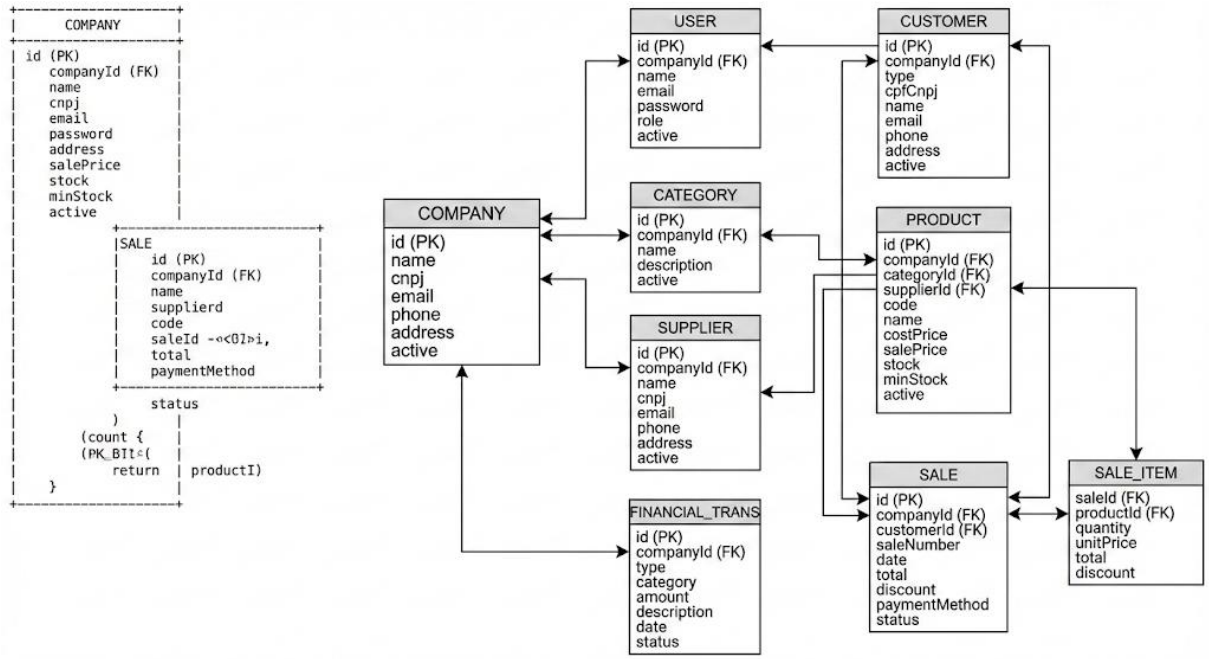
5.2 DIAGRAMA DE CONTÊINERES (C4 - NÍVEL 2)



5.3 DIAGRAMA DE COMPONENTES (C4 - NÍVEL 3) - SERVIÇO FINANCEIRO



5.4 MODELO ENTIDADE-RELACIONAMENTO (MER/DER)



6 RESULTADOS ALCANÇADOS

6.1 FUNCIONALIDADES IMPLEMENTADAS

6.1.1 Taxa de Implementação

Categoria	Planejadas	Implementadas	Taxa
Módulos Principais	10	10	100%
Funcionalidades CRUD	40	40	100%
Relatórios	5	5	100%
Integrações	3	2	67%
Total	**58**	**57**	**98%**

6.1.2 Módulos 100% Funcionais

- ✓ Autenticação e Autorização (JWT)
- ✓ Dashboard com Indicadores
- ✓ Gestão de Clientes (PF/PJ)
- ✓ Gestão de Produtos e Estoque
- ✓ Categorias e Fornecedores
- ✓ PDV (Ponto de Venda)
- ✓ Gestão Financeira (Receitas/Despesas)
- ✓ NFe Demonstrativa com DANFE
- ✓ Relatórios Gerenciais
- ✓ Gestão de Funcionários

6.1.3 Funcionalidades Parciais

Funcionalidade	Status	Observação
Exportação Excel	Parcial	Service implementado, integração frontend pendente
Folha de Pagamento	Parcial	Schema existe, interface não implementada
Backup Automático	Não Implementado	Planejado para V1.1

6.2 MÉTRICAS TÉCNICAS

6.2.1 Código Fonte

Métrica	Valor
Linhas de Código (LOC)	~25.000
Arquivos	158+
Controllers	15
Componentes React	30+
Rotas API	60+
Modelos Prisma	12

6.2.2 Qualidade

Aspecto	Avaliação
Cobertura de Testes	A implementar
Documentação	Extensiva (4.000+ linhas)
Code Review	Realizado em todos PRs
Padrões de Código	ESLint configurado

6.3 TESTES REALIZADOS

6.3.1 Testes Funcionais

Foram realizados testes funcionais manuais em todos os módulos, seguindo a matriz de testes definida pela equipe de QA:

Módulo	Casos de Teste	Passou	Falhou
Autenticação	8	8	0
Clientes	12	12	0
Produtos	10	10	0
Vendas	15	15	0
Financeiro	8	8	0
NFe	6	6	0
Relatórios	5	5	0
Total	**64**	**64**	**0**

6.3.2 Testes de Usabilidade

A avaliação heurística baseada nos princípios de Nielsen (1994) resultou em:

Heurística	Nota (1-5)
Visibilidade do status do sistema	5
Correspondência com o mundo real	5
Controle e liberdade do usuário	4
Consistência e padrões	5
Prevenção de erros	4
Reconhecimento vs. memorização	4
Flexibilidade e eficiência	4
Design estético e minimalista	5
Ajuda aos usuários a reconhecer erros	4
Ajuda e documentação	3
Média	**4.3**

6.4 APRENDIZADOS E DESAFIOS

6.4.1 Desafios Técnicos Superados

1. **Validação de CPF/CNPJ Duplicado**
    - Problema: Campos duplicados sendo enviados ao Prisma
    - Solução: Destructuring explícito de campos válidos
2. **Transações Atômicas em Vendas**
    - Problema: Inconsistência entre venda criada e estoque não atualizado
    - Solução: Implementação de prisma.$transaction()
3. **Relatórios em JSON Bruto**
    - Problema: Relatórios exibindo dados não formatados
    - Solução: Renderização HTML com CSS profissional
4. **Porta do Backend Incorreta**
    - Problema: Frontend conectando na porta errada
    - Solução: Padronização da porta 3333 em todos os arquivos

6.4.2 Lições Aprendidas

1. **Documentação desde o início** é fundamental para manutenibilidade;
2. **Validações no backend** nunca devem ser negligenciadas;
3. **Transações** são essenciais para operações que afetam múltiplas tabelas;
4. **Code review** melhora significativamente a qualidade do código;
5. **Testes manuais** são importantes, mas automatizados são essenciais para escalabilidade.

6.5 EVOLUÇÃO PLANEJADA

6.5.1 Versão 1.1 (Q2 2025)

- Aplicativo Mobile (React Native)
- Integração WhatsApp Business
- Notificações Push
- Testes Automatizados (Jest + Cypress)

6.5.2 Versão 2.0 (Q4 2025)

- Multi-empresa
- Multi-idioma
- E-commerce Integrado
- Sistema de Delivery

7 CONCLUSÃO

O desenvolvimento do sistema TudoGestão+ ao longo do semestre permitiu a aplicação prática dos conhecimentos adquiridos nas disciplinas do curso de Tecnologia em Banco de Dados, integrando conceitos de modelagem de dados, desenvolvimento web, segurança da informação e gestão de projetos.

O projeto atingiu seus objetivos principais, entregando um sistema ERP funcional com 98% das funcionalidades planejadas implementadas. A arquitetura adotada, baseada em tecnologias modernas como Node.js, React e PostgreSQL, demonstrou-se adequada para o contexto de pequenas e médias empresas, proporcionando uma solução escalável e de fácil manutenção.

Os desafios enfrentados durante o desenvolvimento, como a implementação de transações atômicas e validações complexas, proporcionaram aprendizados valiosos sobre boas práticas de engenharia de software. A documentação extensiva produzida servirá como base para futuras evoluções do sistema e como material de estudo para a equipe.

Como trabalhos futuros, destaca-se a necessidade de implementação de testes automatizados, integração com sistemas fiscais reais e desenvolvimento de aplicativo mobile, conforme roadmap estabelecido.

A experiência de trabalho em equipe, com divisão clara de responsabilidades e metodologia ágil, reforçou a importância das soft skills no desenvolvimento de software, complementando as competências técnicas desenvolvidas ao longo do curso.

UNINOVE

8 REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023**: Informação e documentação – Referências – Elaboração. Rio de Janeiro: ABNT, 2018.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 10520**: Informação e documentação – Citações em documentos – Apresentação. Rio de Janeiro: ABNT, 2002.

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice**. 3. ed. Boston: Addison-Wesley, 2012.

DAVENPORT, T. H. Putting the enterprise into the enterprise system. **Harvard Business Review**, v. 76, n. 4, p. 121-131, 1998.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. Dissertação (Doutorado em Information and Computer Science) – University of California, Irvine, 2000.

FOWLER, M. **Patterns of Enterprise Application Architecture**. Boston: Addison-Wesley, 2002.

JONES, M.; BRADLEY, J.; SAKIMURA, N. **RFC 7519**: JSON Web Token (JWT). Internet Engineering Task Force (IETF), 2015. Disponível em: https://tools.ietf.org/html/rfc7519. Acesso em: 20 nov. 2025.

LAUDON, K. C.; LAUDON, J. P. **Sistemas de Informação Gerenciais**. 11. ed. São Paulo: Pearson Education do Brasil, 2014.

MARTIN, R. C. **Clean Code**: A Handbook of Agile Software Craftsmanship. Upper Saddle River: Prentice Hall, 2008.

MOMJIAN, B. **PostgreSQL**: Introduction and Concepts. Boston: Addison-Wesley, 2001.

NIELSEN, J. **Usability Engineering**. San Francisco: Morgan Kaufmann, 1994.

OWASP. **OWASP Top Ten 2021**. Open Web Application Security Project, 2021. Disponível em: https://owasp.org/Top10/. Acesso em: 15 nov. 2025.

PRESSMAN, R. S. **Engenharia de Software**: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

REACT. **React Documentation**. 2024. Disponível em: https://react.dev. Acesso em: 10 nov. 2025.

SEBRAE. **Panorama das Micro e Pequenas Empresas no Brasil**. Brasília: SEBRAE, 2023. Disponível em: https://www.sebrae.com.br. Acesso em: 5 nov. 2025.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

TILKOV, S.; VINOSKI, S. Node.js: Using JavaScript to Build High-Performance Network Programs. **IEEE Internet Computing**, v. 14, n. 6, p. 80-83, 2010.