Documentação Completa do ERP - Eliel Diniz

Visão Geral do Projeto

Este sistema ERP foi desenvolvido com foco em micro e pequenas empresas, oferecendo uma solução integrada para gestão de empresas, clientes, produtos e emissão de Nota Fiscal Eletrônica (NFe). A arquitetura segue boas práticas de separação de responsabilidades, manutenção facilitada e escalabilidade.

Tecnologias Utilizadas

- Node.js + Express para backend
- Prisma ORM com PostgreSQL
- JWT para autenticação segura
- Arquitetura modular em domínios: auth, company, client, product, nfe

---

Arquitetura da Aplicação

A aplicação é dividida em:

- **Controllers:** manipulam requisições e respostas
- **Services:** contêm a lógica de negócio
- **Middlewares:** validações, autenticação e logs
- **Prisma (ORM):** para comunicação com o banco de dados PostgreSQL
- **Utils:** utilitários gerais, como criptografia, validação de documentos, etc.

Módulos Documentados

Autenticação e Segurança

Prompt Utilizado:
Especificar o fluxo completo de autenticação em um ERP: entrada (email/senha), verificação com hash bcrypt, geração de token JWT com expiração, proteção de rotas com middleware, logout e invalidação de sessão.

Resultado:

- Senha armazenada com bcrypt
- JWT gerado e armazenado no frontend
- Middleware `requireAuth` protege rotas privadas

Cadastro de Empresas

Prompt Utilizado:
Descrever o fluxo de cadastro de empresa, incluindo validação de CNPJ, coleta de dados (nome, IE, endereço completo), persistência no banco de dados, tratamento de erros, e resposta ao cliente com confirmação do cadastro e ID gerado.

Resultado:

- Empresa cadastrada com CNPJ validado
- Endereço estruturado aninhado
- Relacionamento com usuários e notas fiscais

Cadastro de Clientes

Prompt Utilizado:
Delinear o fluxo de criação e edição de clientes, contemplando entrada de dados (nome, CPF/CNPJ, endereço), validação fiscal, vinculação à empresa, atualização de histórico e resposta com status da operação.

Resultado:

- Cliente PF ou PJ validado
- Vínculo com a empresa emissora
- Validação de documentos aplicada

Cadastro de Produtos

Prompt Utilizado:
Descrever o fluxo para cadastrar e atualizar produtos: dados básicos e fiscais (NCM, CFOP, tributos), controle de estoque mínimo, verificação de duplicatas, armazenamento e confirmação ao usuário.

Resultado:

- Produtos cadastrados com dados fiscais opcionais
- Estoque controlado com alertas por quantidade

Emissão de NFe

Prompt Utilizado:
Documentar o fluxo completo da emissão de NFe: recebimento do pedido (empresa, cliente e itens), geração do XML conforme schema, validação local (xmllint), assinatura com certificado A1, envio à SEFAZ (homologação ou produção), tratamento de erros e retorno de protocolo.

Resultado:

- XML gerado e validado conforme schema 4.00
- Assinado digitalmente com certificado A1
- Enviado via SOAP para a SEFAZ com protocolo de retorno

Assinatura Digital de XML

Prompt Utilizado:
Especificar o fluxo técnico para ler o certificado (.pem/.pfx), carregar chave privada, aplicar

assinatura digital ao XML da NFe, montar envelope SOAP e transmitir ao endpoint da SEFAZ, com logs detalhados de resposta e erros.

Resultado:

- Assinatura com OpenSSL
- Integração com certificado digital A1 em ambiente de homologação

---

Validação de XML

Prompt Utilizado:
Definir o fluxo para validar localmente o XML da NFe antes do envio: carregamento do arquivo, execução de `xmllint`, análise de erros de schema e resposta estruturada para ajustes no frontend/backend.

Resultado:

- Validado com `xmllint` antes do envio
- Redução de rejeições por schema

Ambiente de Desenvolvimento

Prompt Utilizado:
Criar um guia de configuração do ambiente de desenvolvimento do ERP, explicando como instalar dependências, configurar variáveis de ambiente (ex: JWT_SECRET, DATABASE_URL), e executar a aplicação localmente com `npm run dev`.

Resultado:

- Ambiente local operando com nodemon
- Scripts padronizados em `package.json`

Documentação de API e Endpoints

Prompt Utilizado:
Gerar documentação das rotas da API REST do ERP, listando os endpoints por domínio

(auth, empresa, cliente, produto, nfe), com método HTTP, path, descrição da funcionalidade, payload esperado e resposta retornada.

Resultado:

- Documentação REST separada por domínio
- Referências claras para frontend consumir a API

Tratamento de Erros

Prompt Utilizado:
Criar documentação técnica das mensagens de erro padronizadas utilizadas no ERP, explicando os códigos HTTP retornados (401, 403, 422 etc.), causas comuns e sugestões de correção para cada erro.

Resultado:

- Códigos HTTP padronizados
- Mensagens estruturadas e amigáveis ao frontend

---

1. Refatoração de Código com IA (Node.js)

Objetivo:
Revisar código Node.js de forma crítica, identificar falhas, melhorias e propor refatorações considerando desempenho, escalabilidade, segurança e legibilidade.

Prompt:

Você é um engenheiro de software com 10 anos de experiência em sistemas concorrentes e distribuídos, especializado em Node.js. Seu trabalho é revisar o código a seguir, identificar falhas, pontos de melhoria e propor refatorações. Utilize uma abordagem estruturada, pensando passo a passo, e justifique cada ponto com base nas melhores práticas recomendadas para Node.js, incluindo desempenho, escalabilidade, legibilidade, segurança e manutenção.

Siga esta sequência de raciocínio:

1. Análise Inicial (Skeleton of Thought)
2. Identificação de Problemas e Pontos de Melhoria (Chain of Thought)
3. Exploração de Alternativas (Tree of Thought)
4. Garantia de Consistência (Self-Consistency)

5. Revisão da Sequência e Conclusão

---

2. Decisão Arquitetural

Objetivo:
Definir a arquitetura ideal para o ERP, avaliando opções como monolito modular, microserviços, serverless e ferramentas ORM vs query builder.

Prompt:

Você é um arquiteto de software sênior com foco em aplicações Node.js escaláveis e seguras. Ajude a decidir a melhor arquitetura para um ERP focado em microempresas que inclua autenticação, cadastro de empresas, clientes, produtos e emissão de NFe.

Analise as opções arquiteturais, avaliando prós e contras em termos de escalabilidade, manutenção, segurança e facilidade de testes:

- Monolito modular com camadas (controllers, services, repositories)
- Microserviços separados por domínio
- Serverless functions para módulos específicos
- Uso de ORM (Prisma) vs query builders

Apresente a decisão recomendada com justificativas técnicas.

---

3. Escolha Tecnológica

Objetivo:
Justificar a seleção da stack tecnológica e bibliotecas para garantir maturidade, segurança, performance e integração.

Prompt:

Você é um especialista em tecnologias backend e banco de dados para sistemas ERP. Considere as opções para a stack do projeto: Node.js + Express, banco relacional PostgreSQL com ORM Prisma, autenticação JWT, e geração de NFes via XML assinado digitalmente.

Justifique a escolha de cada tecnologia e biblioteca, considerando:
- Maturidade e comunidade
- Facilidade de integração
- Segurança e performance
- Experiência da equipe

Inclua recomendações para bibliotecas auxiliares essenciais e práticas para garantir qualidade do código e segurança.

---

4. Planejamento e Organização do Projeto

Objetivo:
Planejar sprints e prioridades para entrega rápida do MVP e fases seguintes.

Prompt:

Ajude a estruturar o planejamento de desenvolvimento de um ERP para microempresas,
- Configuração de ambiente e infraestrutura
- Desenvolvimento do módulo de autenticação e autorização
- Cadastro de empresas e clientes
- Cadastro e controle de produtos

Esta documentação consolidada inclui:

- Detalhes do fluxo funcional do ERP
- Principais prompts usados para tomada de decisões técnicas, arquiteturais e refatoração
- Justificativas que demonstram senioridade e boas práticas

Você é um arquiteto de software sênior. Gere um diagrama detalhado e organizado que represente a arquitetura de um sistema ERP para micro e pequenas empresas, incluindo os seguintes módulos e suas interações:

- Autenticação (login, JWT, proteção de rotas)
- Cadastro de Empresas (com validação de CNPJ e dados de endereço)
- Cadastro de Clientes (PF/PJ com validação de documentos)
- Cadastro de Produtos (dados básicos e fiscais, controle de estoque)
- Emissão de Nota Fiscal Eletrônica (NFe) (geração, assinatura digital, envio para SEFAZ)
- Banco de Dados PostgreSQL com tabelas principais e relacionamentos
- Fluxo de dados entre frontend, backend, banco de dados e serviços externos (SEFAZ)

Organize o diagrama em camadas: frontend, backend (com seus módulos), banco de dados e serviços externos. Inclua setas indicando o fluxo de dados e as principais operações entre eles.

Se possível, gere o diagrama no formato Mermaid para fácil visualização e edição.

Prompt de Revisão Passo a Passo (Pré-Refatoração)

(registrado para futuras iterações de código)

Campo	Conteúdo
Título	Revisão por engenheiro sênior – diagnóstico e refatoração guiada
Prompt original	`text Você é um engenheiro de software com 10 anos de experiência em sistemas concorrentes e distribuídos. Seu trabalho é revisar o código a seguir e identificar falhas ou melhorias. Pense passo a passo; justifique cada ponto com base nas práticas recomendadas em Node.js. Ao final, revise a sequência de etapas e forneça uma conclusão objetiva. Use a seguinte estrutura: Etapa 1: <descrição> Etapa 2: <descrição> `
Objetivo	• Fazer **code-review analítico** antes de qualquer refatoração • Gerar lista de problemas e recomendações alinhadas a boas práticas Node.js • Garantir que a refatoração seja fundamentada e rastreável
Estrutura esperada	1. **Etapa 1 – Análise de Arquitetura** 2. **Etapa 2 – Padrões de Código / Qualidade** 3. **Etapa 3 – Performance / Concorrência** 4. **Etapa 4 – Segurança / Validação** 5. **Conclusão Objetiva**
Técnicas implícitas	- Pensamento *step-by-step* (Chain-of-Thought) - Justificativa por boas práticas oficiais (Node.js docs, 12-factor, OWASP) - Foco em **action-items** claros por etapa
Quando aplicar	- Antes de refatorar módulos críticos (API, serviços de fila, workers) - Ao integrar novas dependências de infraestrutura ou padrões de concorrência
Saída esperada	Lista priorizada de falhas, riscos, débitos técnicos e sugestões concretas de melhoria, cada uma com justificativa técnica.
Exemplo de aplicação	- Detectar uso de funções síncronas bloqueantes em rotas Express. - Sugerir pool de conexões no Prisma para evitar *connection storm*. - Apontar falta de `helmet()` e `rate-limit` em pontos de entrada.
Resultado típico	Documento de revisão que serve de roteiro de refatoração e base para *pull request* incremental.