DESCRIÇÃO DO PROJETO

O **Sistema de Gestão Poseidons** tem como objetivo desenvolver uma **aplicação web completa** para gerenciamento de **Clientes, Produtos e Pedidos**, simulando funcionalidades de sistemas ERP, como o **TOTVS Protheus**.
O projeto visa oferecer uma experiência moderna, segura e responsiva, abrangendo desde o **cadastro de clientes e produtos** até o **controle de estoque e pedidos integrados**.

Componentes do Projeto

- **Backend:** Node.js com Express.js, oferecendo uma **API REST** completa para operações CRUD e transações de pedidos.
- **Frontend:** HTML, CSS e JavaScript puro, consumindo a API e exibindo uma interface moderna e responsiva.
- **Banco de Dados:** MySQL, administrado via MySQL Workbench, com tabelas integradas e chaves estrangeiras.

A aplicação permite simular processos empresariais típicos de ERPs, proporcionando uma experiência aprimorada na manipulação de cadastros, controle de estoque e emissão de pedidos — com interface profissional e funcionalidades robustas.

SUMÁRIO

---

1. INTRODUÇÃO

Este projeto integra a disciplina de **Laboratório de Engenharia de Software**, com o objetivo de proporcionar aos alunos **experiência prática em todas as etapas do desenvolvimento de sistemas**, desde o levantamento de requisitos até a entrega de um **MVP (Produto Mínimo Viável)** funcional.

A empresa fictícia **MaicoSoft Importados** foi usada como estudo de caso. Seus colaboradores relataram dificuldades na rotina de cadastros, destacando a **complexidade da interface**, a **dificuldade em localizar campos obrigatórios** e a **falta**

de flexibilidade no sistema atual. Além disso, o diretor solicitou a **redução de custos com licenciamento** e o **envio de notificações automáticas** a cada novo registro.

Diante desse cenário, a equipe desenvolveu uma **aplicação moderna**, baseada em **arquitetura multicamadas (MVC + Serviços + Repositórios)**, com **backend RESTful**, **banco relacional MySQL** e **frontend web responsivo**.
O projeto não apenas atende às necessidades descritas, mas também consolida os conceitos estudados de **engenharia de software**, **modelagem**, **testes**, **desenvolvimento colaborativo** e **documentação técnica**.

---

2. LEVANTAMENTO DE REQUISITOS

2.1 Requisitos Funcionais

Gestão de Clientes (CRUD)

- **RF001 – Cadastrar Cliente**
  Entradas: Dados do cliente (loja, razão social, tipo, etc.)
  Processamento: Geração automática de código sequencial (C00001, C00002…)
  Saída: Confirmação de cadastro
  Critérios: Campos obrigatórios validados, código único, modal interativo.
- **RF002 – Listar Clientes**
  Exibe a lista completa de clientes em tabela responsiva, com carregamento dinâmico.
- **RF003 – Visualizar Cliente**
  Exibição detalhada dos dados em modal somente leitura.
- **RF004 – Atualizar Cliente**
  Permite edição com formulário pré-preenchido e validações. O código não pode ser alterado.
- **RF005 – Excluir Cliente**
  Remove cliente com confirmação obrigatória (ação irreversível).

Gestão de Produtos (CRUD)

- **RF006 – Cadastrar Produto**
  Entradas: nome, categoria, preço, estoque, etc.
  Critérios: Campos obrigatórios validados, controle de estoque mínimo e código único.
- **RF007 – Listar Produtos**
  Exibe todos os produtos com status ativo/inativo e dados organizados em tabela responsiva.
- **RF008 – Atualizar e Excluir Produto**
  Edição e exclusão com feedback visual e confirmação de segurança.

Gestão de Pedidos

- **RF009 – Cadastrar Pedido**
  Entradas: cliente, itens do pedido, quantidades e observações.
  Processamento: Validação de estoque, cálculo automático de totais e baixa automática de itens.
  Saída: Confirmação de criação e atualização de estoque.
- **RF010 – Listar e Visualizar Pedidos**
  Exibe pedidos com dados do cliente, status e valor total.
  Critérios: Relacionamento entre pedidos e itens, status atualizável.
- **RF011 – Cancelar Pedido**
  Exclui pedido, restaurando o estoque automaticamente.

API REST

- **Clientes:**
  `GET /clientes`, `GET /clientes/:codigo`, `POST /clientes`, `PUT /clientes/:codigo`, `DELETE /clientes/:codigo`
- **Produtos:**
  `GET /produtos`, `GET /produtos/:codigo`, `POST /produtos`, `PUT /produtos/:codigo`, `DELETE /produtos/:codigo`
- **Pedidos:**
  `GET /pedidos`, `GET /pedidos/:codigo`, `POST /pedidos`, `PUT /pedidos/:codigo`, `DELETE /pedidos/:codigo`

Validação de Dados

- Campos obrigatórios e validação em tempo real.
- Mensagens de erro padronizadas via middleware `AppError`.
- Regras aplicadas tanto no **frontend** quanto no **backend**.

2.2 Requisitos Não Funcionais

- **Performance:** Tempo de resposta < 2 segundos.
- **Usabilidade:** Interface intuitiva, feedback visual e *loading states*.
- **Responsividade:** Layout adaptável a desktop, tablet e mobile.
- **Confiabilidade:** Tratamento robusto de erros e logs detalhados.
- **Segurança:** CORS configurado, códigos únicos e não editáveis.

---

3. BACKLOG DO PRODUTO

Prioridade	ID	História de Usuário	Critério de Aceitação
Alta	US01	Formulário intuitivo para cadastro de cliente	Validação de campos e confirmação
Alta	US02	Persistência de dados de clientes no banco	Dados gravados corretamente
Alta	US03	Visualização da lista de clientes e produtos	Listagem funcional e ordenada
Alta	US04	CRUD completo de produtos	Operações GET, POST, PUT, DELETE
Alta	US05	Cadastro e controle de pedidos	Estoque atualizado e cálculo automático
Média	US06	Layout responsivo	Interface adaptável
Média	US07	Redução de custos de licenciamento	Independente de sistemas proprietários
Baixa	US08	Documentação técnica completa	Contém arquitetura, endpoints e uso
Baixa	US09	Testes automatizados básicos	Testes unitários e integração

---

4. ARQUITETURA E TECNOLOGIAS

Padrão Utilizado:

MVC + Serviços + Repositórios

Backend:

- Node.js + Express.js
- MySQL2 (acesso ao banco via pool de conexões)
- CORS e dotenv configurados
- Tratamento centralizado de erros (`AppError` + middleware)

Banco de Dados:

- MySQL 8.0+
- Tabelas: `clientes`, `produtos`, `pedidos`, `pedido_itens`
- Relacionamentos e chaves estrangeiras

Frontend:

- HTML5, CSS3 (Flexbox, Grid, gradientes e animações)

- JavaScript (ES6+) com Fetch API
- Modais interativos e navegação dinâmica

Ferramentas de Apoio:

- Git/GitHub, VS Code, Postman, MySQL Workbench

---

5. DESENVOLVIMENTO

O backend foi totalmente refatorado para seguir o padrão **MVC**, dividido em:

- **Controllers:** Recebem requisições e retornam respostas padronizadas.
- **Services:** Implementam regras de negócio, validações e cálculos.
- **Repositories:** Realizam consultas SQL isoladas.
- **Error Handling:** Middleware global de tratamento de exceções.

O frontend foi desenvolvido com **HTML, CSS e JS puro**, oferecendo uma interface fluida, responsiva e intuitiva.

---

6. TESTES

- **Testes de API:** realizados via Postman (CRUD completo).
- **Testes de UI:** fluxos de cadastro, atualização e exclusão.
- **Validação de Campos:** frontend e backend.
- **Logs:** implementados para depuração e rastreabilidade.

---

7. INOVAÇÕES

- Arquitetura **MVC** modular e escalável.
- **Gestão integrada** de Clientes, Produtos e Pedidos.
- **Baixa automática de estoque** ao registrar pedidos.
- **Tratamento de erros centralizado.**
- **Validação em tempo real** no frontend e backend.
- Interface moderna e responsiva, com **gradientes e animações suaves**.
- **Geração automática de códigos** (C00001, P00001, PED00001).

---

8. DIÁRIO DE BORDO

Informações do Projeto:

- Projeto: Sistema de Gestão Poseidons
- Disciplina: Laboratório de Engenharia de Software
- Equipe: João Pedro Amaral, Otavio Correa, Samuel Siqueira e Vinicius Bueno

Atividades:

- **Semana 1:** Planejamento, modelagem do banco e implementação das rotas iniciais.
- **Semana 2:** Criação da interface, integração com API e design responsivo.
- **Semana 3:** Implementação completa do CRUD e módulo de pedidos, testes e documentação.

---

9. CONCLUSÃO

O **Sistema de Gestão Poseidons** representa uma implementação completa dos conceitos de **engenharia de software moderna**, evidenciando domínio em desenvolvimento **full-stack**, **arquitetura MVC**, **metodologias ágeis** e **boas práticas de documentação**.

A aplicação atende integralmente aos objetivos propostos, oferecendo uma base sólida para expansões futuras e demonstrando a capacidade técnica da equipe em desenvolver sistemas empresariais completos.

🔗 **GitHub:** https://github.com/OtavioCa04/poseidons