Nome do Projeto

Stock Manager Web – Sistema Web de Controle de Estoque para Pequenas Empresas.

. Objetivo

Desenvolver um sistema web simples e intuitivo para auxiliar pequenas empresas no **controle de produtos**, **registro de entradas e saídas**, **alertas de estoque baixo** e **relatórios de vendas**.

3. Requisitos

Requisitos Funcionais (RF)

1. **RF01** – Cadastrar produtos (nome, categoria, preço, quantidade em estoque).
2. **RF02** – Editar e excluir produtos.
3. **RF03** – Registrar entrada e saída de produtos.
4. **RF04** – Gerar relatórios de estoque e movimentações.
5. **RF05** – Alertar quando o estoque estiver abaixo de um valor mínimo.
6. **RF06** – Autenticação de usuários (login e senha).

Requisitos Não Funcionais (RNF)

1. **RNF01** – Interface responsiva (funcionar em PC, tablet e celular).
2. **RNF02** – Segurança no armazenamento de senhas (hash).
3. **RNF03** – Tempo de resposta inferior a 2 segundos por requisição.

4. Tecnologias

- **Front-end**: HTML5, CSS3, JavaScript, Bootstrap.
- **Back-end**: Node.js com Express.
- **Banco de Dados**: MySQL.
- **Hospedagem**: Heroku .
- **Controle de Versão**: Git/GitHub.

5. Modelagem UML

Diagrama de Casos d**e Uso** (simplificado):

- Ator: **Administrador**
  - Cadastrar Produto
  - Editar Produto
  - Registrar Entrada/Saída
  - Gerar Relatório
  - Receber Alerta

7. Estrutura de Banco de Dados (MySQL)

MySQL

```sql
CREATE TABLE usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    senha_hash VARCHAR(255) NOT NULL
);

CREATE TABLE produtos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,S
    categoria VARCHAR(50),
    preco DECIMAL(10,2),
    quantidade INT DEFAULT 0,
    estoque_minimo INT DEFAULT 0


CREATE TABLE movimentacoes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    produto_id INT NOT NULL,
    tipo ENUM('entrada', 'saida') NOT NULL,
    quantidade INT NOT NULL,
    data_movimentacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (produto_id) REFERENCES produtos(id)
```

8. Exemplo de Código (Node.js + Express – Cadastro de Produto)

Javascript

```
const express = require('express');
const mysql = require('mysql2');
const app = express();S
app.use(express.json());

const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',S
    password: '',
    database: 'estoque'



app.post('/produtos', (req, res) => {
    const { nome, categoria, preco, quantidade, estoque_minimo } = req.body;
    db.query(
        'INSERT INTO produtos (nome, categoria, preco, quantidade, estoque_minimo) VALUES (?, ?, ?, ?, ?)',
        [nome, categoria, preco, quantidade, estoque_minimo],
        (err) => {
            if (err) return res.status(500).json({ error: err });
            res.status(201).json({ message: 'Produto cadastrado com sucesso!' });




app.listen(3000, () => console.log('Servidor rodando na porta 3000'));
```

Stock Manager Web V1

```
// stockmanager_web_v1 - Sistema Web de Controle de Estoque (versão inicial)

// Estrutura: Node.js + Express + MySQL (versão simples, sem autenticação ainda)

const express = require('express');
const mysql = require('mysql2');
const cors = require('cors');

const app = express();
const port = 3000;

app.use(cors());
app.use(express.json());

// Conexão com o banco de dados
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'estoque'


db.connect(err => {
  if (err) {
    console.error('Erro ao conectar no banco de dados:', err);
  } else {
    console.log('Conectado ao banco de dados MySQL.');
```

```
// Cadastrar produto
app.post('/produtos', (req, res) => {
  const { nome, categoria, preco, quantidade, estoque_minimo } = req.body;
  const sql = 'INSERT INTO produtos (nome, categoria, preco, quantidade, estoque_minimo) VALUES (?, ?, ?, ?, ?)';
  db.query(sql, [nome, categoria, preco, quantidade, estoque_minimo], (err, result) => {
    if (err) return res.status(500).json({ erro: err });
    res.status(201).json({ mensagem: 'Produto cadastrado com sucesso!' });


// Listar todos os produtos
app.get('/produtos', (req, res) => {
  db.query('SELECT * FROM produtos', (err, resultados) => {
    if (err) return res.status(500).json({ erro: err });
    res.status(200).json(resultados);


// Atualizar produto
app.put('/produtos/:id', (req, res) => {
  const { id } = req.params;
  const { nome, categoria, preco, quantidade, estoque_minimo } = req.body;
  const sql = 'UPDATE produtos SET nome = ?, categoria = ?, preco = ?, quantidade = ?, estoque_minimo = ? WHERE id =
  db.query(sql, [nome, categoria, preco, quantidade, estoque_minimo, id], (err, result) => {
    if (err) return res.status(500).json({ erro: err });
    res.json({ mensagem: 'Produto atualizado com sucesso!' });

// Excluir produto
app.delete('/produtos/:id', (req, res) => {
  const { id } = req.params;
  db.query('DELETE FROM produtos WHERE id = ?', [id], (err, result) => {
    if (err) return res.status(500).json({ erro: err });
    res.json({ mensagem: 'Produto excluído com sucesso!' });


// Iniciar servidor
app.listen(port, () => {
  console.log(`Servidor rodando em http://localhost:${port}`);
```