**Método para a Construção de um Sistema Web Responsivo para Gestão de Produtos**

**Objetivo**

Desenvolvido um sistema web responsivo para cadastro, listagem e exclusão de produtos com persistência de dados em um banco de dados SQLite, utilizou API Node.js para o back-end e HTML, CSS e JavaScript para o front-end.

Materiais Necessários

Software e Ferramentas:

Anaconda Prompt – Versão: conda 24.4.0

Node.js – Versão: v20.13.1

npm – Versão: 10.5.2

Editor de código :Sublime build 4169

Navegador web (Google Chrome, Brave, Edge)

**Linguagens de Programação e Tecnologias:**

HTML5, CSS3, JavaScript

Node.js com Express

SQLite – Versão: 3.41.2 2023-03-22 11:56:21 0d1fc92f94cb6b76bffe3ec34d69cffde2924203304e8ffc4155597af0c191da

Biblioteca body-parser para parseamento de requisições HTTP

Biblioteca cors para habilitar CORS (Cross-Origin Resource Sharing)

**Passos**

**1. Preparação do Ambiente**

**1.1 Instalação do Node.js e npm:**

Realizamos o download e instalação do Node.js a partir do site oficial (https://nodejs.org/). Verificamos a instalação executando os comandos no Anaconda Prompt (https://www.anaconda.com/download):

**node -v**

**npm -v**

**1.2 Criação do Diretório do Projeto**

Criamos e navegamos até o diretório do projeto:  
**mkdir tememcasa**

**cd tememcasa**

**1.3 Inicialização do Projeto Node.js**

Inicializamos um novo projeto Node.js com o comando

**npm init -y**

**Configuração do Servidor Node.js com Express e SQLite**

**2.1 Criação do Arquivo server.js**

Crie um arquivo **server.js** na raiz do projeto com o seguinte conteúdo:

const express = require('express');

const sqlite3 = require('sqlite3').verbose();

const bodyParser = require('body-parser');

const cors = require('cors');

const path = require('path');

const app = express();

const port = 3000;

app.use(bodyParser.json());

app.use(cors());

app.use(express.static(path.join(\_\_dirname, 'public')));

// Criação ou conexão com o banco de dados SQLite

const db = new sqlite3.Database(':memory:');

db.serialize(() => {

db.run(`CREATE TABLE IF NOT EXISTS produtos (

id INTEGER PRIMARY KEY AUTOINCREMENT,

numero INTEGER,

produto TEXT,

tipo TEXT,

quantidade REAL,

preco REAL,

total REAL

)`);

});

// Rota para servir o arquivo HTML

app.get('/', (req, res) => {

res.sendFile(path.join(\_\_dirname, 'index.html'));

});

// Rota para adicionar produto

app.post('/api/produtos', (req, res) => {

const { numero, produto, tipo, quantidade, preco, total } = req.body;

const stmt = db.prepare(`INSERT INTO produtos (numero, produto, tipo, quantidade, preco, total) VALUES (?, ?, ?, ?, ?, ?)`);

stmt.run(numero, produto, tipo, quantidade, preco, total, function(err) {

if (err) {

return res.status(500).json({ error: err.message });

}

res.json({ id: this.lastID });

});

stmt.finalize();

});

// Rota para listar produtos

app.get('/api/produtos', (req, res) => {

db.all(`SELECT \* FROM produtos`, [], (err, rows) => {

if (err) {

return res.status(500).json({ error: err.message });

}

res.json({ produtos: rows });

});

});

// Rota para deletar produto

app.delete('/api/produtos/:id', (req, res) => {

const id = req.params.id;

db.run(`DELETE FROM produtos WHERE id = ?`, id, function(err) {

if (err) {

return res.status(500).json({ error: err.message });

}

res.json({ deletedID: id });

});

});

app.listen(port, () => {

console.log(`Servidor rodando em http://localhost:${port}`);

});

**3. Criação da Interface do Usuário**

**3.1 Estrutura do Projeto**

Organizamos o projeto conforme a estrutura:

tememcasa/

│

├── public/

│ ├── css/

│ │ └── styles.css

│ ├── js/

│ │ └── script.js

│ └── images/

│ └── img-home.svg

│

├── server.js

└── index.html

**3.2 Criação do Arquivo HTML**

Criamos um arquivo **index.html** com o seguinte conteúdo:

<!DOCTYPE html>

<html lang="pt-BR">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Página Inicial - TemEmCasa</title>

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap" rel="stylesheet">

<link rel="stylesheet" href="/css/styles.css">

</head>

<body>

<div class="container">

<div class="header">

<h1 class="title">TemEmCasa</h1>

<div class="title-with-icon">

<img src="/images/img-home.svg" alt="Página Inicial">

</div>

<p class="welcome-message">Seja bem-vindo ao TemEmCasa! Aqui você pode organizar sua despensa e sua lista de compras de forma prática e eficiente.</p>

</div>

<h2>Cadastro de Produtos</h2>

<form id="productForm">

<input type="number" id="numero" placeholder="Número" required>

<input type="text" id="produto" placeholder="Produto" required>

<input type="number" id="quantidade" placeholder="Quantidade" required>

<select id="tipo" required>

<option value="">Volume</option>

<option value="kg">kg</option>

<option value="litro">litro</option>

<option value="grama">grama</option>

<option value="pacote">pacote</option>

<option value="lata">lata</option>

<option value="embalagem">embalagem</option>

<option value="duzia">dúzia</option>

<option value="unidade">unidade</option>

</select>

<input type="number" id="preco" placeholder="Preço" step="0.01" required>

<button type="button" onclick="addProduct()">Adicionar Produto</button>

</form>

<h3>Lista de Produtos</h3>

<table id="productTable">

<thead>

<tr>

<th>Número</th>

<th>Produto</th>

<th>Tipo</th>

<th>Quantidade</th>

<th>Preço</th>

<th>Total</th>

<th>Ação</th>

</tr>

</thead>

<tbody>

</tbody>

<tfoot>

<tr>

<td colspan="4"></td>

<th>Soma:</th>

<td id="totalSum"></td>

<td></td>

</tr>

</tfoot>

</table>

</div>

<script src="/js/script.js"></script>

</body>

</html>

**3.3 Criação do Arquivo de Estilos CSS:**

Criamos o arquivo public/css/styles.css com o seguinte conteúdo:

/\* Reset de estilos \*/

body {

font-family: 'Poppins', sans-serif;

background-color: #f4f4f9;

color: #496989;

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

flex-direction: column;

}

/\* Estilos de seções \*/

.container {

width: 90%;

max-width: 700px;

margin: 30px auto;

padding: 30px 20px;

background-color: #496989;

color: #fff;

box-shadow: 0 0 15px rgba(0, 0, 0, 0.2);

border-radius: 10px;

text-align: center;

}

.header {

display: flex;

align-items: center;

justify-content: center;

flex-direction: column;

margin-bottom: 20px;

}

.title-with-icon {

display: flex;

align-items: center;

justify-content: center;

margin-bottom: 10px;

}

.welcome-message {

text-align: center;

margin-top: 10px;

font-size: 1.1em;

}

form {

margin-bottom: 20px;

}

input, select, button {

padding: 10px;

margin: 10px 0;

border: none;

border-radius: 20px;

display: block;

width: 90%;

max-width: 300px;

margin-left: auto;

margin-right: auto;

box-sizing: border-box;

background-color: #d3d3d3;

color: #58a399;

font-size: 1em;

font-family: 'Poppins', sans-serif;

}

::placeholder {

color: #58A399;

}

button {

background-color: #58A399;

color: #fff;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover {

background-color: #ffc107;

}

table {

width: 100%;

border-collapse: collapse;

margin: 20px 0;

background-color: #58A399;

color: #fff;

border-radius: 10px;

overflow: hidden;

}

th, td {

padding: 10px;

border-bottom: 2px solid #fff;

text-align: center;

}

th {

background-color: #496989;

}

img {

max-width: 300px;

height: auto;

display: block;

margin: 20px auto;

}

img:hover {

transform: scale(1.1);

transition: transform 0.3s ease;

}

/\* Estilos de animação e hover \*/

@keyframes fadeIn {

from {

opacity: 0;

transform: translateY(-20px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

h1, h2 {

animation: fadeIn 1s ease forwards;

}

/\* Estilos do título \*/

.title {

font-size: 2.5em;

background: linear-gradient(45deg, #58A399, #FFC107);

-webkit-background-clip: text;

color: transparent;

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.2);

background-size: 200% 200%;

animation: colorChange 3s infinite alternate;

}

@keyframes colorChange {

from {

background-position: 0% 50%;

}

to {

background-position: 100% 50%;

}

}

/\* Media Queries para Responsividade \*/

@media (max-width: 600px) {

.container {

padding: 20px 10px;

}

h1, h2 {

font-size: 1.5em;

}

.title {

font-size: 2em;

}

input, select, button {

width: 100%;

}

img {

max-width: 200px;

}

table {

width: 100%;

margin: 10px 0;

background-color: #496989; /\* Ajuste a cor de fundo \*/

color: #fff;

border-radius: 10px;

overflow: hidden;

}

th, td {

padding: 5px;

text-align: center;

}

}

**3.4 Criação do Arquivo JavaScript:**

Criamos o arquivo public/js/script.js com o seguinte conteúdo:  
  
async function addProduct() {

const numero = document.getElementById('numero').value;

const produto = document.getElementById('produto').value;

const tipo = document.getElementById('tipo').value;

const quantidade = parseFloat(document.getElementById('quantidade').value);

const preco = parseFloat(document.getElementById('preco').value).toFixed(2);

const total = (quantidade \* preco).toFixed(2);

const produtoData = { numero, produto, tipo, quantidade, preco, total };

const response = await fetch('http://localhost:3000/api/produtos', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(produtoData)

});

if (response.ok) {

loadFromDatabase();

document.getElementById('productForm').reset();

} else {

alert('Erro ao adicionar produto!');

}

}

async function deleteProduct(id) {

const response = await fetch(`http://localhost:3000/api/produtos/${id}`, {

method: 'DELETE'

});

if (response.ok) {

loadFromDatabase();

} else {

alert('Erro ao excluir produto!');

}

}

async function loadFromDatabase() {

const response = await fetch('http://localhost:3000/api/produtos');

const data = await response.json();

const produtos = data.produtos;

const table = document.getElementById('productTable').getElementsByTagName('tbody')[0];

table.innerHTML = ''; // Limpar tabela antes de carregar novos dados

produtos.forEach(function(produto) {

const row = table.insertRow();

row.insertCell(0).textContent = produto.numero;

row.insertCell(1).textContent = produto.produto;

row.insertCell(2).textContent = produto.tipo;

row.insertCell(3).textContent = produto.quantidade;

row.insertCell(4).textContent = produto.preco;

row.insertCell(5).textContent = produto.total;

const deleteCell = row.insertCell(6);

deleteCell.innerHTML = `<button onclick="deleteProduct(${produto.id})">Excluir</button>`;

});

// Atualiza o total geral ao carregar os dados

calculateTotal();

}

function calculateTotal() {

const table = document.getElementById('productTable').getElementsByTagName('tbody')[0];

const rows = table.rows;

let sum = 0;

for (let i = 0; i < rows.length; i++) {

const total = parseFloat(rows[i].cells[5].textContent);

sum += total;

}

document.getElementById('totalSum').textContent = sum.toFixed(2);

}

document.addEventListener('DOMContentLoaded', loadFromDatabase);

**4. Iniciando o Servidor e Verificando a Aplicação**

**4.1 Iniciar o Servidor Node.js:**

Executamos o comando para iniciar o servidor:  
  
**node server.js**

**4.2 Abrir a Aplicação no Navegador:**

Ao abrimos o navegador e acesse **http://localhost:3000**.

Verificamos que a aplicação está funcionando corretamente, incluindo a adição, listagem e exclusão de produtos.

Esses foram os passos, para obter um sistema web completo que permite o cadastro, listagem e exclusão de produtos, com persistência de dados em um banco de dados SQLite e uma interface responsiva utilizando HTML, CSS e JavaScript.