📚 Guia Completo: Projeto Base para Apresentação

Projeto: **Sistema de Arrecadação Centralizada** (Node.js, Express, MongoDB) - Otimizado para Interação Local.

🛠️ Passo 0: Preparação do Ambiente e Dependências

Execute estes comandos na pasta raiz do projeto para criar o `package.json` e instalar as bibliotecas necessárias:

1. Inicialização e Instalação:

```
npm init -y
npm install express mongoose
```

O `express` é o framework de servidor. O `mongoose` conecta e gerencia o MongoDB.

💻 Backend: O Servidor Central (`server.js`)

Este código implementa as Peças 1, 2, 3 e 4. Ele inicia o servidor, conecta ao banco e define as rotas de leitura e escrita.

```
// ====================================================
// IMPORTS E CONEXÃO (Peça 2)
// ====================================================
const express = require('express');
const mongoose = require('mongoose');
const path = require('path');

const app = express();
const PORT = 3000;
const DB_URI = 'mongodb://localhost:27017/projetoArrecadacao';

mongoose.connect(DB_URI)
    .then(() => console.log('✅ Conectado ao MongoDB!'))
    .catch(err => console.error('❌ Erro de conexão:', err));

// Schema (Formato do dado - um único campo 'total')
const arrecadacaoSchema = new mongoose.Schema({
    total: { type: Number, required: true, default: 0 },
    nome: { type: String, required: true }
});
const Arrecadacao = mongoose.model('Arrecadacao', arrecadacaoSchema);

// Garante que o registro 'CaixaGeral' exista
async function inicializarCaixaGeral() {
    const caixaExiste = await Arrecadacao.findOne({ nome: 'CaixaGeral' });
    if (!caixaExiste) {
        await new Arrecadacao({ nome: 'CaixaGeral', total: 0 }).save();
        console.log('📦 Registro inicial criado.');
    }
}
inicializarCaixaGeral();

// ====================================================
// MIDDLEWARES E CONFIGURAÇÃO (Peça 1)
// ====================================================

// Habilita a leitura de JSON e de dados de formulários HTML
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Serve arquivos da pasta 'public'
app.use(express.static(path.join(__dirname, 'public')));


// ====================================================
// ROTAS DO PROJETO (Peças 3 e 4)
// ====================================================

// ROTA 1: LEITURA (GET /total - Peça 3)
app.get('/total', async (req, res) => {
    try {
        const caixaGeral = await Arrecadacao.findOne({ nome: 'CaixaGeral' });
        res.json({ total: caixaGeral ? caixaGeral.total : 0 });
    } catch (error) {
        res.status(500).json({ erro: 'Erro ao buscar total.' });
    }
});
```

```
// ROTA 2: ESCRITA (POST /depositar - Peca 4)
app.post('/depositar', async (req, res) => {
    try {
        const valorDeposito = parseFloat(req.body.valor);

        if (typeof valorDeposito !== 'number' || valorDeposito <= 0 || isNaN(valorDeposito)) {
            return res.status(400).send('Valor inválido.');
        }

        // Usa o operador atômico $inc para garantir que a soma seja segura
        await Arrecadacao.findOneAndUpdate(
            { nome: 'CaixaGeral' },
            { $inc: { total: valorDeposito } },
            { new: true, upsert: true }
        );

        // Chave da simplicidade: Redireciona para o formulário após o POST (refresh)
        res.redirect('/depositar.html?status=sucesso');

    } catch (error) {
        res.status(500).send('Erro interno ao depositar.');
    }
});


// ====================================================
// INICIALIZACÃO
// ====================================================
app.listen(PORT, '0.0.0.0', () => {
    console.log(`Servidor rodando em http://localhost:${PORT}`);
});
```

🖥️ Frontend: A Interface (Pasta `public`)

Crie uma pasta chamada **`public`** na raiz do projeto, e dentro dela, os arquivos HTML e o CSS.

1. Tela do Aluno (Visualização): `public/display.html`

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Tela de Apresentação</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container display-container">
        <h1>🌟 Total Arrecadado ao Vivo</h1>
        <div class="card">
            <h2>VALOR ATUAL:</h2>
            <div id="total-arrecadado" class="total-value">Buscando...</div>
        </div>
        <p class="refresh-info">Atualização automática a cada 5 segundos.</p>
    </div>

    <script>
        // Função para ligar o Back ao Front (GET)
        const totalDisplay = document.getElementById('total-arrecadado');
        const BASE_URL = window.location.origin;

        async function carregarTotal() {
            try {
                // response: o pacote HTTP; data: o objeto JS utilizável
                const response = await fetch(`${BASE_URL}/total`);
                const data = await response.json();

                if (response.ok) {
                    totalDisplay.textContent = `R$ ${data.total.toLocaleString('pt-BR', { minimumFractionDigits: 2
                }
            } catch (error) {
                totalDisplay.textContent = 'Erro de conexão.';
            }
        }

        carregarTotal();
        // Auto-Refresh a cada 5 segundos
        setInterval(carregarTotal, 5000);
    </script>
</body>
</html>
```

2. Tela do Interator (Depósito): `public/depositar.html`

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Interface de Depósito</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container form-container">
        <h1>➡️ Faça sua Contribuição</h1>

        <!-- Usa formulário HTML nativo para o POST -->
        <form action="/depositar" method="POST">
            <label for="valor">Quanto você quer depositar?</label>
            <input type="number" name="valor" id="valor" required min="1" value="10">
            <button type="submit">DEPOSITAR</button>
        </form>

        <p class="refresh-info">A tela de apresentação do aluno atualizará em instantes!</p>
    </div>
    <script>
        // Alerta de sucesso após o redirecionamento do servidor
        const urlParams = new URLSearchParams(window.location.search);
        if (urlParams.get('status') === 'sucesso') {
            alert("Depósito realizado com sucesso!");
        }
    </script>
</body>
</html>
```

3. Estilos Base: `public/style.css` (Para impressão)

Use o CSS para estilizar e destacar os valores.

```
body {
    font-family: 'Arial', sans-serif;
    color: #333;
    padding: 20px;
    margin: 0;
    min-height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #fff;
}
.container {
    background-color: #f8f8f8;
    padding: 30px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    max-width: 500px;
    width: 100%;
    text-align: center;
}
h1 {
    color: #007bff;
}
.total-value {
    font-size: 3.5em;
    color: #28a745;
    font-weight: bold;
    margin: 15px 0;
}
input[type="number"] {
    padding: 12px;
    border: 1px solid #ccc;
    border-radius: 4px;
    font-size: 1.1em;
    width: 100%;
    box-sizing: border-box;
    margin-bottom: 10px;
}
button {
    padding: 12px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 1.2em;
}
.refresh-info {
    font-size: 0.9em;
    color: #888;
    margin-top: 20px;
}
/* Estilo para impressão */
@media print {
    body {
        box-shadow: none;
```

```
    }
}
```