**API Farmácia com FrontEnd**

por: github.com/iurisaints

Arquitetura do projeto utilizando a convenção normal:

apifarmacia/ (pasta raiz)

\*\*|-- node\_modules/\*\*

\*\*|-- src/

| |— clients/\*\*

| |— |— clientes.js

\*\*| |— controllers/\*\*

| |— |— controllerClientes.js

\*\*| |— |— data/\*\*

| |— |— |— dadosClientes.json

\*\*| |— telas/\*\*

| |— |— telaClientes.js

| |-- app.js (seu servidor Node.js)

|-- package.json

|-- ...

Antes de tudo, precisamos configurar nosso front:

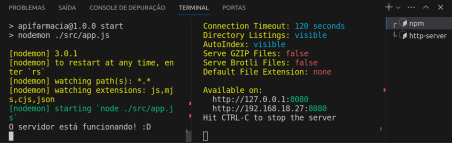
no terminal:

npm install -g http-server

e após isso só digitar *http-server* no terminal.

Esse comando tem que estar funcionando paralelamente ao *npm start.*

Exemplo:



Client:

Esse tipo de script incorporado em uma página HTML, que interage com uma API backend usando JavaScript, é frequentemente referido como "JavaScript Cliente" ou "Cliente da API". Ele é responsável por realizar solicitações HTTP para o backend, processar respostas e manipular a interface do usuário de acordo.

Exemplo nesse caso:

// clientes.js

document.addEventListener('DOMContentLoaded', function () {

// Carregar a lista de clientes ao carregar a página

loadClientesList();

// Adicionar um ouvinte de evento ao formulário para adicionar clientes

document.getElementById('formAdicionarCliente').addEventListener('submit', function (event) {

event.preventDefault();

adicionarCliente();

});

});

API Farmácia com FrontEnd 1

function adicionarCliente() {

const nome = document.getElementById('nomeCliente').value;

const endereco = document.getElementById('enderecoCliente').value;

const email = document.getElementById('emailCliente').value;

const telefone = document.getElementById('telefoneCliente').value;

fetch('http://localhost:3000/api/clientes', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

nome: nome,

endereco: endereco,

email: email,

telefone: telefone,

}),

})

.then(response => response.json())

.then(data => {

console.log(data);

loadClientesList(); // Recarregar a lista após adicionar um cliente

})

.catch(error => console.error('Error:', error));

}

function loadClientesList() {

fetch('http://localhost:3000/api/clientes')

.then(response => response.json())

.then(data => displayClientesList(data))

.catch(error => console.error('Error:', error));

}

function displayClientesList(data) {

const listaClientes = document.getElementById('listaClientes');

listaClientes.innerHTML = '';

data.forEach(cliente => {

const listItem = document.createElement('li');

listItem.textContent = `Nome: ${cliente.nome} - Endereço: ${cliente.endereco} - Email: ${cliente.email} - Telefone: ${cliente.t listaClientes.appendChild(listItem);

});

}

Controllers:

São as partes do código que lidam diretamente com a manipulação de dados (como leitura/gravação de arquivos JSON), elas geralmente são chamadas de "serviços" ou "controladores" em vez de "servidores", para evitar confusões com o conceito tradicional de servidor HTTP.

Exemplo nesse caso:

const express = require('express');

const server = express();

const dadosClientes = require('./data/dadosClientes.json'); // Novo arquivo para dados de clientes

const fs = require('fs');

server.use(express.json());

server.post('/clientes', (req, res) => {

const novoCliente = req.body;

if (!novoCliente.nome || !novoCliente.endereco || !novoCliente.email || !novoCliente.telefone) {

return res.status(400).json({ mensagem: "Dados incompletos, tente novamente" });

} else {

dadosClientes.Cliente.push(novoCliente);

salvarDadosClientes(dadosClientes);

return res.status(201).json({ mensagem: "Novo cliente cadastrado com sucesso!" });

}

});

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

return res.json(dadosClientes.Cliente);

});

server.put('/clientes/:id', (req, res) => {

const clienteId = parseInt(req.params.id);

const atualizarCliente = req.body;

const idCliente = dadosClientes.Cliente.findIndex(c => c.id === clienteId);

API Farmácia com FrontEnd 2

if (idCliente === -1) {

return res.status(404).json({ mensagem: "Cliente não encontrado :/" });

} else {

dadosClientes.Cliente[idCliente].nome = atualizarCliente.nome || dadosClientes.Cliente[idCliente].nome; dadosClientes.Cliente[idCliente].endereco = atualizarCliente.endereco || dadosClientes.Cliente[idCliente].endereco; dadosClientes.Cliente[idCliente].email = atualizarCliente.email || dadosClientes.Cliente[idCliente].email; dadosClientes.Cliente[idCliente].telefone = atualizarCliente.telefone || dadosClientes.Cliente[idCliente].telefone;

salvarDadosClientes(dadosClientes);

return res.json({ mensagem: "Cliente atualizado com sucesso!" });

}

});

server.delete("/clientes/:id", (req, res) => {

const clienteId = parseInt(req.params.id);

dadosClientes.Cliente = dadosClientes.Cliente.filter(c => c.id !== clienteId);

salvarDadosClientes(dadosClientes);

return res.status(200).json({ mensagem: "Cliente excluído com sucesso" });

});

function salvarDadosClientes() {

fs.writeFileSync(\_\_dirname + '/data/dadosClientes.json', JSON.stringify(dadosClientes, null, 2)); }

module.exports = { server, salvarDadosClientes };

Data:

Onde temos nosso banco de dados.

{

"Cliente": [

{

"nome": "Iuri",

"endereco": "Rua dos Andrades",

"email": "iurisaints@gmail.com",

"telefone": "55996840281"

},

{

"nome": "Rafael Bittencourt",

"endereco": "Rua da Música, 123",

"email": "rbiteca@gmail.com",

"telefone": "559996840281"

}

]

}

Telas:

Onde se encontram as telas HTML.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Gerenciamento de Clientes</title>

<style>

label {

display: block;

margin-top: 10px;

}

</style>

</head>

<body>

<h1>Gerenciamento de Clientes</h1>

<h2>Adicionar Cliente</h2>

<form id="formAdicionarCliente">

<label for="nomeCliente">Nome:</label>

<input type="text" id="nomeCliente" name="nomeCliente" required>

<label for="enderecoCliente">Endereço:</label>

<input type="text" id="enderecoCliente" name="enderecoCliente" required>

<label for="emailCliente">Email:</label>

<input type="email" id="emailCliente" name="emailCliente" required>

API Farmácia com FrontEnd 3

<label for="telefoneCliente">Telefone:</label>

<input type="text" id="telefoneCliente" name="telefoneCliente" required>

<button type="button" onclick="adicionarCliente()">Adicionar</button> </form>

<h2>Listar Clientes</h2>

<ul id="listaClientes"></ul>

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

</body>

</html>

Demais códigos:

app.js:

const express = require('express');

const medicamentosRouter = require('./controllers/controllerMedicamentos'); const fornecedoresRouter = require('./controllers/controllerFornecedores'); const vendasRouter = require('./controllers/controllerVendas');

const clientesRouter = require('./controllers/controllerClientes'); const cors = require('cors');

const app = express();

app.use(cors()); // Adicione esta linha para configurar o CORS

app.use('/api', medicamentosRouter.server); // '/api/medicamentos' app.use('/api', fornecedoresRouter.server); // '/api/fornecedores' app.use('/api', vendasRouter.server); // '/api/vendas'

app.use('/api', clientesRouter.server); // '/api/clientes'

app.listen(3000, () => {

console.log('O servidor está funcionando! :D');

});

package.json:

{

"dependencies": {

"cors": "^2.8.5",

"express": "^4.18.2",

"nodemon": "^3.0.1"

},

"name": "apifarmacia",

"version": "1.0.0",

"main": "./src/app.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "nodemon ./src/app.js"

},

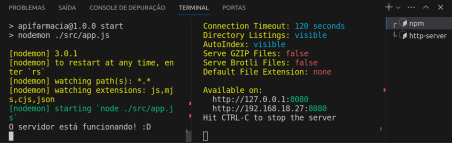
"author": "",

"license": "ISC",

"description": ""

}

API Farmácia com FrontEnd 4

API Farmácia com FrontEnd 5