</> DESVENDANDO O CÓDIGO

SUA PRIMEIRA API REST COM NODE.JS E EXPRESS

GUIA PRÁTICO GRATUITO

```
const express = require("express");
const app = express();

app.get("/", (req, res) => {
  res.json({ status: "online" });
});
```

PROFESSOR MARCOS CALDAS

Do primeiro servidor à visão de um projeto completo

COMECE POR AQUI

O backend deixa de ser uma “caixa-preta”

Este guia foi criado para quem já conhece um pouco de JavaScript e quer entender o que acontece quando uma aplicação envia ou recebe dados.

RESULTADO DESTE GUIA

Você criará uma API pequena com Node.js e Express, entenderá rotas e métodos HTTP e visualizará como essa base evolui até um projeto integrado com frontend, ViaCEP e MongoDB.

O que você vai aprender

- O papel de uma API REST.
- Como iniciar um servidor Express.
- Como criar rotas GET e POST.
- Como receber e validar JSON.
- Como testar respostas e códigos HTTP.
- Qual é o próximo passo para construir um CRUD profissional.

CONCEITO ESSENCIAL

O caminho de uma requisição

Quando o usuário realiza uma ação, o frontend envia uma requisição. O backend interpreta o pedido, executa uma regra e devolve uma resposta.

1. **FRONTEND**
   Envia dados ou solicita uma informação.
2. **ROTA**
   Reconhece a URL e o método HTTP.
3. **REGRA**
   Processa, valida ou consulta outro serviço.
4. **RESPOSTA**
   Devolve status e JSON ao cliente.

PENSE ASSIM

A rota é a porta. O método HTTP informa a intenção. O JSON transporta os dados. O código de status comunica o resultado.

Os quatro métodos do CRUD

GET LER	**Consultar dados** Exemplo: GET /produtos	**STATUS** **200**
POST CRIAR	**Cadastrar um recurso** Exemplo: POST /produtos	**STATUS** **201**
PUT ATUALIZAR	**Modificar um recurso** Exemplo: PUT /produtos/1	**STATUS** **200**
DELETE EXCLUIR	**Remover um recurso** Exemplo: DELETE /produtos/1	**STATUS** **204**

CRUD significa Create, Read, Update e Delete. Ele é uma base de raciocínio, não um limite: sistemas reais também possuem autenticação, filtros, regras, integrações e relatórios.

MÃO NO CÓDIGO

Criando o primeiro servidor Express

Crie uma pasta, abra o terminal nela e execute:

TERMINAL

```
mkdir primeira-api
cd primeira-api
npm init -y
npm install express
```

Agora crie o arquivo server.js:

SERVER.JS

```
const express = require("express");
const app = express();

app.use(express.json());

app.get("/", (req, res) => {
  res.status(200).json({ mensagem: "API funcionando" });
});

app.listen(3000, () => {
  console.log("Servidor em http://localhost:3000");
});
```

Execute node server.js e acesse http://localhost:3000. Se aparecer o JSON, sua primeira rota está funcionando.

ROTAS NA PRÁTICA

GET para consultar dados

Antes de conectar um banco, podemos entender a lógica usando um array em memória.

LISTAGEM

```
const produtos = [
  { id: 1, nome: "Teclado", preco: 120 },
  { id: 2, nome: "Mouse", preco: 80 }
];

app.get("/produtos", (req, res) => {
  res.status(200).json(produtos);
});
```

Experimente

- Reinicie o servidor.
- Acesse /produtos no navegador.
- Adicione um terceiro produto ao array.
- Observe que os dados somem quando o servidor reinicia: ainda não existe persistência.

PRÓXIMA EVOLUÇÃO

Trocar o array em memória por MongoDB e organizar a aplicação em route, controller, service e model.

POST, JSON e validação

A rota POST recebe dados no body. A API deve validar a entrada antes de criar o recurso.

CRIAÇÃO

```
app.post("/produtos", (req, res) => {
  const { nome, preco } = req.body;

  if (!nome || typeof preco !== "number") {
    return res.status(400).json({
      erro: "Informe nome e preço válido"
    });
  }

  const produto = { id: Date.now(), nome, preco };
  produtos.push(produto);
  return res.status(201).json(produto);
});
```

Teste com este JSON

```
{
  "nome": "Monitor",
  "preco": 899
}
```

O return interrompe a função depois da resposta. O status 400 sinaliza entrada inválida; o status 201 informa que um recurso foi criado.

LEITURA PROFISSIONAL

O código de status também é informação

200

OK

Consulta ou atualização concluída.

201

CREATED

Novo recurso criado.

400

BAD REQUEST

Entrada ou regra inválida.

404

NOT FOUND

Recurso não encontrado.

500

SERVER ERROR

Falha inesperada no servidor.

FAZER FUNCIONAR NÃO É O FIM

Uma API previsível combina dados corretos, status correto e mensagens claras. Isso facilita o frontend, os testes e a manutenção.

PROJETO REAL

Do formulário ao ViaCEP

No projeto do curso, o aluno parte de um frontend em HTML, CSS e JavaScript e evolui para uma aplicação com backend próprio.

O guia gratuito apresenta a porta de entrada. No curso completo, essa arquitetura é construída passo a passo, com CRUD, organização em camadas, validação, erros e persistência.

HORA DE TESTAR

Checklist da sua primeira API

- [ ] O servidor inicia sem erros.
- [ ] GET / retorna status 200.
- [ ] GET /produtos devolve uma lista.
- [ ] POST /produtos aceita um JSON válido.
- [ ] POST inválido devolve status 400.
- [ ] A resposta usa application/json.
- [ ] Cada rota tem uma responsabilidade clara.
- [ ] Nenhuma senha aparece no código ou na resposta.

DESAFIO

Crie a rota GET /produtos/:id. Se o produto não existir, responda 404 com uma mensagem clara.

O que continua no curso completo

Este guia entregou uma API funcional e a base de raciocínio. O curso transforma essa base em uma aplicação organizada e conectada.

- CRUD completo com GET, POST, PUT e DELETE.
- Rotas, controllers, services e models.
- Validação e tratamento centralizado de erros.
- MongoDB e Mongoose para persistência.
- Testes organizados das rotas.
- Frontend do projeto ViaCEP.
- Integração entre frontend, backend, ViaCEP e banco.
- Projeto final preparado para portfólio.

TRANSFORMAÇÃO

Sair de uma API simples e chegar a um CRUD organizado, testável e conectado ao banco, compreendendo o caminho completo dos dados.

</> DESVENDANDO O CÓDIGO

VOCÊ CRIOU A BASE. AGORA CONSTRUA A APLICAÇÃO COMPLETA.

Continue no curso API REST com Node.js e Express e transforme fundamentos em um projeto completo com frontend, backend, ViaCEP e MongoDB.

CONHEÇA O CURSO COMPLETO

desvendandoocodigo.com.br

PROFESSOR MARCOS CALDAS

Professor de Desenvolvimento de Sistemas

Conteúdo prático, projetos e evolução passo a passo

MATERIAL GRATUITO • COMPARTILHE COM QUEM ESTÁ COMEÇANDO