

HTTP e Rotas

REVISÃO

O que é o HTTP?

- Hypertext Transfer Protocol;
- Protocolo para a transferência de dados na internet, permitindo a comunicação entre navegadores e servidores web;
- Cada requisição HTTP:
 - Método (GET, POST, etc.);
 - URL (endereço do recurso);
 - Cabeçalhos;
 - Corpo (body): usado em POST/PUT

Métodos HTTP

- GET
 - Pede informações ao servidor;
 - Dados enviados na URL (?chave=valor).
- POST
 - Envia dados no corpo da requisição;
 - Usado para criar novos recursos;
 - Dados **não** aparecem na URL.
- PUT
 - Atualiza recurso existente;
 - Ex: alterar dados de um produto cadastrado.
- DELETE
 - Remove recurso;
 - Ex: Excluir um usuário.

Código no express

```
app.get('/teste', (req, res) => {  
  res.json({ metodo: 'GET', query: req.query });  
});
```

```
app.post('/teste', (req, res) => {  
  res.json({ metodo: 'POST', body: req.body });  
});
```

```
app.put('/teste', (req, res) => {  
  res.json({ metodo: 'PUT', body: req.body });  
});
```

```
app.delete('/teste', (req, res) => {  
  res.json({ metodo: 'DELETE', query: req.query });  
});
```

Principais propriedades do req

- Sobre a requisição:
 - req.method (get, post, etc)
 - req.url (caminho + query)
 - req.originalUrl (url original mesmo com middleware)
 - req.path (sem a query)
 - req.protocol (http ou https)
 - req.secure (true p/ https)
- Parâmetros:
 - req.params (rota dinâmica)
 - req.query (query string)
 - req.body (precisa do middleware)

Principais propriedades do req

- Cabeçalho:
 - req.headers (todo cabeçalho)
 - req.get('Header-Name') – cabeçalho específico
- Sobre a rede:
 - req.hostname
 - req.ip
 - req.ips
- Outras úteis:
 - req.cookies (cookies enviados)
 - req.signedCookies (cookies assinados)

Principais propriedades do res

- Enviar resposta:
 - `res.send(body)`: envia txt, html
 - `res.json(obj)`: formato json
 - `res.status (code)`: status http (200, 404, etc)
- Cabeçalhos da resposta:
 - `res.set(campo, valor)`: define o cabeçalho
 - `res.get(campo)`: recupera valor de um cabeçalho
 - `res.type(mimeType)`: atalho para definir o Content-Type
 - `res.type('json').send({ ok: true });`
 - `res.type('html').send("Olá!");`

Principais propriedades do **res**

- Redirecionamento:
 - `res.redirect(url)`
- Arquivo:
 - `res.download(caminho)`
 - `res.sendFile(caminho)`
- Cookies:
 - `res.cookie (nome, valor, opções)`
 - `res.clearCookie (nome)`

Testando com *curl*

```
curl -X GET "http://localhost:3000/teste?nome=Ana"
```

```
curl -X POST http://localhost:3000/teste \  
-H "Content-Type: application/json" \  
-d '{"nome":"Ana"}'
```

```
curl -X PUT http://localhost:3000/teste \  
-H "Content-Type: application/json" \  
-d '{"id":10,"status":"ativo"}'
```

```
curl -X DELETE "http://localhost:3000/teste?id=123"
```

Middleware

- `express.json()`
 - Serve para interpretar o corpo da requisição (`req.body`) quando ele vem no formato JSON.
 - Se não tiver `express.json()`, o `req.body` vem como `undefined`.
- `express.urlencoded()`
 - Serve para interpretar o corpo da requisição (`req.body`) quando ele vem no formato padrão de formulários HTML).

`extended: true`

```
{
  usuario: {
    nome: "Ana",
    idade: "22"
  }
}
```

`extended: false`

```
{ "usuario[nome]": "Ana", "usuario[idade]": "22" }
```

Resumo

- HTTP é a base da comunicação WEB.
 - GET: buscar;
 - POST: enviar;
 - PUT: atualizar;
 - DELETE: excluir.
- Diferença fundamental: query string x corpo da requisição.

ROTAS NO EXPRESS

ROTAS ESTÁTICAS

Rotas estáticas - Conceito

- O caminho não muda
- Sempre respondem para o mesmo endereço (URL) – não possuem variáveis e nem parâmetros
- Servem para páginas fixas

Rotas estáticas - Estrutura

```
app.get('/rota', (req, res) => {  
  res.send('Resposta da rota');  
});
```

- app: Servidor Express;
- get: método HTTP;
- '/rota': caminho fixo da URL;
- (req,res) => { } : função callback, que será chamada sempre que o cliente acessa esta rota

Rotas estáticas - Exemplos

```
app.get('/home', (req, res) => {  
  res.send('Bem-vindos!');  
});
```

```
app.get('/contato', (req, res) => {  
  res.send('<h1>Contatos</h1>');  
});
```

```
app.get('/sobre', (req, res) => {  
  res.json({  
    disciplina: "Desenvolvimento Web"  
    curso: : "Informática"  
  });  
});
```

ROTAS DINÂMICAS

Rotas dinâmicas com Parâmetros (req.params)

- Possuem variáveis na URL que mudam a resposta
- Essas variáveis são chamadas de parâmetros de rota
- No Express, parâmetros de rota são definidos com dois pontos : no caminho

```
app.get('/produto/:id', (req, res) => { ... })
```

Rotas dinâmicas - Estrutura

```
app.get('/rota/:parametro', (req, res) => {  
  res.send('Parâmetro recebido:  
  ${req.params.parametro} ');  
});
```

- :parametro: é um *placeholder*, que será substituído pelo valor enviado na URL
- req.params: objeto que contém todos os parâmetros da rota

Rotas dinâmicas - Exemplo

```
app.get('/usuario/:id', (req, res) => {  
  res.send('Usuário solicitado:  
    ${req.params.id}');  
});
```

– /usuario/12 -> “Usuário solicitado: 12”

Rotas dinâmicas - Exemplo

```
app.get('/pedido/:pedidoId/item/:itemId', (req,  
res) => {  
  const { pedidoId, itemId } = req.params;  
  res.json({ pedidoId, itemId });  
});
```

- /pedido/5/item/99 → { "pedidoId": "5", "itemId":
 "99" }

Rotas dinâmicas - Exemplo

```
app.get('/blog/:slug/:comentariold?', (req, res) => {  
  res.json({  
    artigo: req.params.slug,  
    comentario: req.params.comentariold ?? "nenhum"  
  });  
});
```

- /blog/nodejs → { artigo: "nodejs", comentario: "nenhum" }
- blog/nodejs/45 → { artigo: "nodejs", comentario: "45" }

Validando parâmetros

```
app.get('/usuario/:id', (req, res) => {  
  const id = Number(req.params.id);  
  if (Number.isNaN(id)) {  
    return res.status(400).json({ erro: "ID deve ser numérico" });  
  }  
  res.json({ id, nome: "Usuário Exemplo" });  
});
```

- /usuario/7 → { id: 7, nome: "Usuário Exemplo" }
- /usuario/abc → { erro: "ID deve ser numérico" } (status 400)

Diferença para Query String

- Parâmetros de rota (req.params): parte do caminho fixo da URL.
 - Ex.: /produto/:id → /produto/10
- Query string (req.query): informações adicionais, após ?.
 - Ex.: /buscar?nome=Caneta

Resumo

- Rotas dinâmicas usam : nome no caminho.
- Os valores vêm em req.params como strings.
- É possível ter múltiplos parâmetros, opcionais e até validar tipos.
- Use params para identificar recursos → produto, usuário, pedido

QUERY STRING

Query String (req.query)

- É parte da URL que vem depois do símbolo ?
- É formada por pares chave=valor, separados por &
- Estrutura:
 - /rota?chave=valor&outraChave=outroValor

```
app.get('/buscar', (req, res) => {  
  const { nome } = req.query; // captura query string  
});
```

Query String (req.query)

- Tudo que vier depois do ? fica disponível no req.query
 - /buscar?nome=caneta&categoria=escolar
 - { nome: "caneta", categoria: "escolar" }

ATIVIDADE

EXEMPLO PRÁTICO (FORMULÁRIO VIA GET E POST)

Estrutura do Projeto

meu-projeto/

└─ public/ ← onde ficam os HTML, CSS e JS

| └─ index.html

| └─ form.html

└─ server.js ← servidor Express

└─ package.json

Configuração do Servidor

- Configurar o servidor
- Definir a porta
- Servidor arquivos estáticos na pasta *public*
- Definir as rotas

Páginas .html

- Crie um arquivo form.html que possua dois forms, um com método get e outro com método post

Testando

1. Rodar o servidor
2. Abrir o navegador
 1. <http://localhost:3000/form.html>
3. Testar
 1. Enviar o formulário GET
 2. Enviar o formulário POST

Observações

- `express.static("public")` → libera os arquivos dentro da pasta public direto pelo navegador.
 - public/index.html
 - <http://localhost:3000/>
 - public/form.html
 - <http://localhost:3000/form.html>