Linguagem de Programação Visual

RICARDO HENDGES

CRUD

MÉTODOS DE REQUISIÇÃO HTTP

O protocolo HTTP define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada para um dado recurso. Embora esses métodos possam ser descritos como substantivos, eles também são comumente referenciados como HTTP Verbs (Verbos HTTP). Cada um deles implementa uma semântica diferente, mas alguns recursos são compartilhados por um grupo deles

GET - O método GET solicita a representação de um recurso específico. Requisições utilizando o método GET devem retornar apenas dados.
POST - O método POST é utilizado para submeter uma entidade a um recurso específico, frequentemente causando uma mudança no estado do recurso ou efeitos colaterais no servidor.
PUT - O método PUT substitui todas as atuais representações do recurso de destino pela carga de dados da requisição.
DELETE - O método DELETE remove um recurso específico.
PATCH - O método PATCH é utilizado para aplicar modificações parciais em um recurso.

CRUD

STATUS DE RESPOSTAS HTTP

Os códigos de status das respostas HTTP indicam se uma requisição HTTP foi corretamente concluída. As respostas são agrupadas em cinco classes:

1. Respostas de informação (100-199)
2. Respostas de sucesso (200-299)
3. Redirecionamentos (300-399)
4. Erros do cliente (400-499)
5. Erros do servidor (500-599)

CRUD

STATUS DE RESPOSTAS HTTP

Respostas de informação (100-199)

100 Continue - Indica que tudo ocorreu bem até agora e que o cliente deve continuar com a requisição ou ignorar se já concluiu o que gostaria.
102 Processing - Indica que o servidor recebeu e está processando a requisição, mas nenhuma resposta está disponível ainda.

CRUD

STATUS DE RESPOSTAS HTTP

Respostas de sucesso (200-299)

200 OK - Estas requisição foi bem sucedida. O significado do sucesso varia de acordo com o método HTTP.

> *O resultado de sucesso de um **PUT** ou **DELETE** geralmente não são **200 OK**, e sim **204 No Content**...*
>
> *Ou **201 Created** quando o recurso é carregado pela primeira vez.*

201 Created *- A requisição foi bem sucedida e um novo recurso foi criado como resultado. Esta é uma típica resposta enviada após uma requisição POST.*

202 Accepted *- A requisição foi recebida mas nenhuma ação foi tomada sobre ela.*

204 No Content *- Não há conteúdo para enviar para esta solicitação.*

CRUD

STATUS DE RESPOSTAS HTTP

Redirecionamentos (300-399)

300 Multiple Choice - A requisição tem mais de uma resposta possível. User-agent ou o user deve escolher uma delas. Não há maneira padrão para escolher uma das respostas.
301 Moved Permanently - Esse código de resposta significa que a URI do recurso requerido mudou. Provavelmente, a nova URI será especificada na resposta.
302 Found - Esse código de resposta significa que a URI do recurso requerido foi mudada temporariamente. Novas mudanças na URI poderão ser feitas no futuro. Portanto, a mesma URI deve ser usada pelo cliente em requisições futuras.
303 See Other - O servidor manda essa resposta para instruir ao cliente buscar o recurso requisitado em outra URI com uma requisição GET.

CRUD

STATUS DE RESPOSTAS HTTP

Erros do cliente (400-499)

400 Bad Request - Essa resposta significa que o servidor não entendeu a requisição pois está com uma sintaxe inválida.
401 Unauthorized - Embora o padrão HTTP especifique "unauthorized", semanticamente, essa resposta significa "unauthenticated". Ou seja, o cliente deve se autenticar para obter a resposta solicitada.
403 Forbidden - O cliente não tem direitos de acesso ao conteúdo portanto o servidor está rejeitando dar a resposta. Diferente do código 401, aqui a identidade do cliente é conhecida.
404 Not Found - O servidor não pode encontrar o recurso solicitado. Este código de resposta talvez seja o mais famoso devido à frequência com que acontece na web.

418 I'm a teapot - O servidor recusa a tentativa de coar café num bule de chá.

CRUD

STATUS DE RESPOSTAS HTTP

Erros do servidor (500-599)

500 Internal Server Error - O servidor encontrou uma situação com a qual não sabe lidar.
501 Not Implemented - O método da requisição não é suportado pelo servidor e não pode ser manipulado.
502 Bad Gateway - Esta resposta de erro significa que o servidor, ao trabalhar como um gateway a fim de obter uma resposta necessária para manipular a requisição, obteve uma resposta inválida.
504 Gateway Timeout - Esta resposta de erro é dada quando o servidor está atuando como um gateway e não obtém uma resposta a tempo.

CRUD

GET - POST (X)
PUT - DELETE - PATCH ( )

```
const alunoController = require('../controllers/aluno');

module.exports = (app) => {
    app.get('/aluno', alunoController.getAlunos)
    app.post('/aluno', alunoController.postAlunos)
}       You, seconds ago • Uncommitted changes
```

CRUD

GET - POST (X)
DELETE (X)
PUT - PATCH ( )

Service

```
const sql_delete =
    `delete from alunos
     where id = $1 `
const deleteAlunos = async (params) => {
    const { id } = params
    await db.query(sql_delete, [id])
}

module.exports.deleteAlunos = deleteAlunos
```

Controller

```
const deleteAlunos = async (req, res, next) => {
    try {
        await alunoService.deleteAlunos(req.params)
            .then(ret => res.status(204).send(ret))
            .catch(err => res.status(500).send(err))
    } catch (err) {
        next(err);
    }
}
module.exports.deleteAlunos = deleteAlunos
```

Router final

```
const alunoController = require('../controllers/aluno');

module.exports = (app) => {
    app.get('/aluno', alunoController.getAlunos)
    app.post('/aluno', alunoController.postAlunos)
    app.delete('/aluno/:id', alunoController.deleteAlunos)
}
```

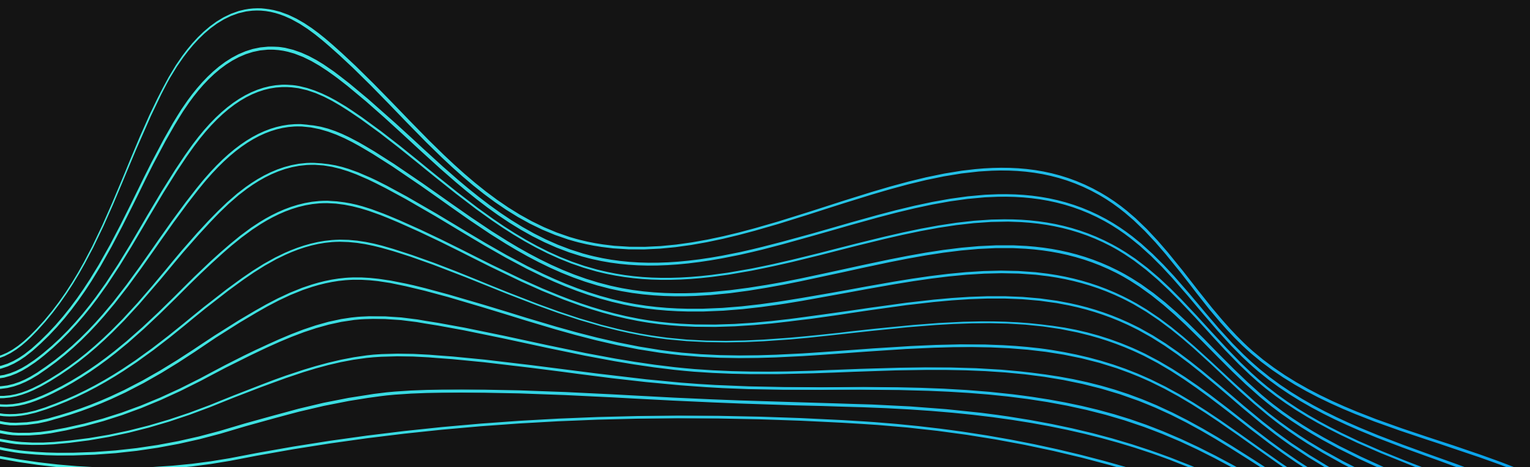
CRUD

GET - POST - DELETE (X)
PUT - PATCH (X)

Em poucas palavras, os métodos HTTP PUT e PATCH são usados para indicar um requisição de alteração de dados.

Geralmente, ao usar-se o PUT, fica legível que a alteração do dado será com referência a entidade completa.
Exemplo: PUT /usuario/1234
Resultado: {'id': 1234, 'name': 'Joao', 'idade': 25, 'documento': '123.321.12-X'}

O PATCH é usado para atualização parcial, quando você não quer mandar o payload completo.
Exemplo: PATCH /usuario/1234
Resultado: {'name': 'João'}



CRUD

GET - POST - DELETE (X)
PUT - PATCH (X)

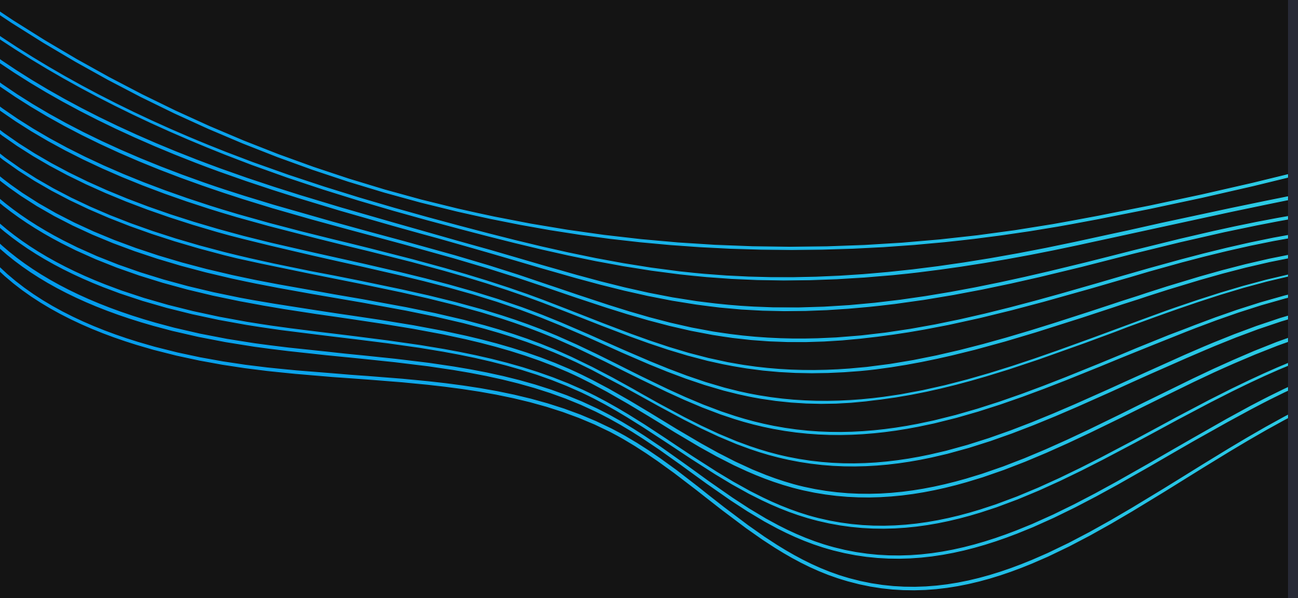
Router final

```
const alunoController = require('../controllers/aluno');

module.exports = (app) => {
    app.get('/aluno', alunoController.getAlunos)
    app.post('/aluno', alunoController.postAlunos)
    app.delete('/aluno/:id', alunoController.deleteAlunos)
    app.put('/aluno/:id', alunoController.putAlunos)
    app.patch('/aluno/:id', alunoController.patchAlunos)
}
```

CRUD

GET - POST - DELETE (X)
PUT - PATCH (X)



Controller final

```
const putAlunos = async (req, res, next) => {
    try {
        let params = req.body
        params.id = req.params.id
        await alunoService.putAlunos(params)
            .then(ret => res.status(200).send(ret))
            .catch(err => res.status(500).send(err))
    } catch (err) {
        next(err);
    }
}

const patchAlunos = async (req, res, next) => {
    try {
        let params = req.body
        params.id = req.params.id
        await alunoService.patchAlunos(params)
            .then(ret => res.status(200).send(ret))
            .catch(err => res.status(500).send(err))
    } catch (err) {
        next(err);
    }
}

module.exports.putAlunos = putAlunos
module.exports.patchAlunos = patchAlunos
```

CRUD

GET - POST - DELETE (X)
PUT - PATCH (X)

Service Put

```
const sql_put =
    `update alunos
        set nome = $2,
            sobrenome = $3,
            periodo = $4,
            observacao = $5
      where id = $1 `
const putAlunos = async (params) => {
    const { id, nome, sobrenome, periodo, observacao } = params
    return await db.query(sql_put, [id, nome, sobrenome, periodo, observacao])
}
```

CRUD

GET - POST - DELETE (X)
PUT - PATCH (X)

Service Patch

Add exports no service

```
module.exports.putAlunos = putAlunos
module.exports.patchAlunos = patchAlunos
```

```
const sql_patch =
    `update alunos
        set `
const patchAlunos = async (params) => {
    let fields = ''
    let binds = []
    binds.push(params.id)
    let countParams = 1
    if (params.nome) {
        countParams ++
        fields += ` nome = $${countParams} `
        binds.push(params.nome)      You, 4 minutes ago • Novos metod
    }
    if (params.sobrenome) {
        countParams ++
        fields += (fields?',':'') + ` sobrenome = $${countParams} `
        binds.push(params.sobrenome)
    }
    if (params.periodo) {
        countParams ++
        fields += (fields?',':'') + ` periodo = $${countParams} `
        binds.push(params.periodo)
    }
    if (params.observacao) {
        countParams ++
        fields += (fields?',':'') + ` observacao = $${countParams} `
        binds.push(params.observacao)
    }
    let sql = sql_patch + fields + ' where id = $1 '
    return await db.query(sql, binds)
}
```

Testando rotas

POST - http://localhost:3000/aluno {inserir 1 aluno}
POST - http://localhost:3000/aluno {inserir 2 aluno}
GET - http://localhost:3000/aluno {visualizar alunos}
DELETE - http://localhost:3000/aluno/1 {deleta 1 aluno}
GET - http://localhost:3000/aluno {visualizar alunos}
PUT - http://localhost:3000/aluno/2 {atualiza tudo do 2 aluno}
GET - http://localhost:3000/aluno {visualizar alunos}
PATCH - http://localhost:3000/aluno/1 {atualiza algo 1 aluno}
GET - http://localhost:3000/aluno {visualizar alunos}