Documentação API RESTful

Renan Sales Ribeiro Pool

Professor Rommel Vieira

Março/2024

Subindo a aplicação

No diretório raíz, devemos executar o seguinte comando:

docker build -t api_node .

Para subir o container na porta 80, devemos executar o comando:

docker run -p 80:3000 api_node

Exemplos de consulta

busca de produtos

- **request**

```
$ curl -v http://localhost/produtos/
*   Trying 127.0.0.1:80...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /produtos/ HTTP/1.1
> Host: localhost
> User-Agent: curl/7.68.0
> Accept: */*
>
```

- **response**

```
< HTTP/1.1 200 OK
< X-Powered-By: Express
< ETag: b40b5fe133420bea64c92c6804c0119c0e5ea393
< Content-Type: application/json; charset=utf-8
< Content-Length: 363
< Date: Tue, 05 Mar 2024 20:50:20 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
<
* Connection #0 to host localhost left intact
[{"id":1,"descricao":"Arroz parboilizado 5Kg","valor":25,"marca":"Tio
João"},{"id":2,"descricao":"Maionese
250gr","valor":7.2,"marca":"Helmans"},{"id":3,"descricao":"Iogurte Natural
200ml","valor":2.5,"marca":"Itambé"},{"id":4,"descricao":"Batata Maior Palha
300gr","valor":15.2,"marca":"Chipps"},{"id":5,"descricao":"Nescau
400gr","valor":8,"marca":"Nestlé"}]
```

busca de produto específico

- **request**

```
$ curl -v http://localhost/produtos/1
*   Trying 127.0.0.1:80...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /produtos/1 HTTP/1.1
> Host: localhost
> User-Agent: curl/7.68.0
> Accept: */*
>
```

- **response**

```
< HTTP/1.1 200 OK
< X-Powered-By: Express
< ETag: b40b5fe133420bea64c92c6804c0119c0e5ea393
< Content-Type: application/json; charset=utf-8
< Content-Length: 363
< Date: Tue, 05 Mar 2024 20:50:20 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
<
* Connection #0 to host localhost left intact
[{"id":1,"descricao":"Arroz parboilizado 5Kg","valor":25,"marca":"Tio João"}]
```

Incluir novo produto

- **request**

```
$ curl -v -X POST   -H "Content-Type: application/json"   -d '{
    "descricao": "chocolate 90g",
    "valor": 6.99,
    "marca": "Garoto"
  }'   http://localhost:80/produtos
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 127.0.0.1:80...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 80 (#0)
> POST /produtos HTTP/1.1
> Host: localhost
```

```
> User-Agent: curl/7.68.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 71
>
```

- **response**

```
< HTTP/1.1 201 Created
< X-Powered-By: Express
< ETag: 97938e88b30206f8d6bdc03d8bc9d56b0bd20aa4
< Content-Type: application/json; charset=utf-8
< Content-Length: 109
< Date: Tue, 05 Mar 2024 20:50:00 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
<
* Connection #0 to host localhost left intact
{"message":"Produto criado com sucesso!","product":{"id":6,"descricao":"chocolate
90g","valor":6.99,"marca":"Garoto"}}
```

modificar produto existente

- **request**

```
$ curl -X PUT -H "Content-Type: application/json" -d '{
    "descricao": "pao",
    "valor": 10.99,
    "marca": "Panco"
}' http://localhost:80/produtos/6
```

- **response**

```
{"message":"Produto atualizado com
sucesso!","product":{"id":6,"descricao":"pao","valor":10.99,"marca":"Nova marca"}}
```

Deletar produto existente

- **request**

```
$ curl -X DELETE http://localhost:80/produtos/6 -v
*   Trying 127.0.0.1:80...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 80 (#0)
> DELETE /produtos/6 HTTP/1.1
> Host: localhost
> User-Agent: curl/7.68.0
> Accept: */*
>
```

- **response**

```
< HTTP/1.1 204 No Content
< X-Powered-By: Express
< ETag: 9c346a7a9963339833a69e1d061266a6adaa9bbd
< Date: Tue, 05 Mar 2024 20:50:16 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
<
* Connection #0 to host localhost left intact
```

Documentação do código

```
Importando o módulo Express
const express = require('express');

Criando uma instância do Express
const app = express();

Definindo o cache da aplicação (Tive dificuldades para implementar o
cache, pois não consegui configurar a automatização da limpeza de cache
após alteração da ETag)
// const apicache = require('apicache');
// const cache = apicache.middleware;
// app.use(cache('5 minutes'));

Definindo o corpo da requisição (body-parser)
app.use(express.json());

Definindo os dados de produtos
const lista_produtos = {
```

```
    produtos: [
        { id: 1, descricao: "Arroz parboilizado 5Kg", valor: 25.00,
marca: "Tio João" },
        { id: 2, descricao: "Maionese 250gr", valor: 7.20, marca:
"Helmans" },
        { id: 3, descricao: "Iogurte Natural 200ml", valor: 2.50,
marca: "Itambé" },
        { id: 4, descricao: "Batata Maior Palha 300gr", valor: 15.20,
marca: "Chipps" },
        { id: 5, descricao: "Nescau 400gr", valor: 8.00, marca:
"Nestlé" },
    ]
}

const crypto = require('crypto');

Rota para obter todos os produtos
app.get('/produtos', /*cache('5 minutes')*/ (req, res) => {
    let hash =
crypto.createHash('sha1').update(JSON.stringify(lista_produtos.produtos
)).digest('hex');
    Verificar se há um cabeçalho If-None-Match na requisição
    if (req.headers['If-None-Match'] === hash) {
        res.status(304).send();
    } else {
        Se os dados foram modificados, atualizar o cabeçalho ETag e
retornar os dados
        res.setHeader('ETag', hash);
        return res.status(200).json(lista_produtos.produtos);
    }
});

 Rota para obter um produto específico
app.get('/produtos/:id', (req, res) => {
    const id = parseInt(req.params.id);
    const produto = lista_produtos.produtos.find(prod => prod.id ===
id);
    if (!produto) {
        return res.status(404).json({ mensagem: 'Produto não
encontrado' });
    }
    res.status(200).json(produto);
});
```

```
Rota para incluir um novo produto
let idCounter = 6;

app.post('/produtos', (req, res) => {
    const novoProduto = req.body;

    Atribua o valor atual do contador ao ID do novo produto
    novoProduto.id = idCounter;

    Crie um novo objeto de produto com a estrutura de dados adequada
    const newProduct = {
      id: idCounter,
      descricao: novoProduto.descricao,
      valor: novoProduto.valor,
      marca: novoProduto.marca,
    };

    Adicione o novo produto à sua lista (substitua
`lista_produtos.produtos` pelo seu array real)
    lista_produtos.produtos.push(newProduct);

    Incremente o contador para o próximo produto
    idCounter++;

    Atualize o hash para refletir a adição do novo produto
    const hash =
crypto.createHash('sha1').update(JSON.stringify(lista_produtos)).digest
('hex');

    Limpe o cache para garantir que a próxima solicitação obtenha os
dados atualizados
    // apicache.clear('/produtos');

    res.setHeader('ETag', hash);

    res.status(201).json({
      message: 'Produto criado com sucesso!',
      product: newProduct,
    });
  });

 Rota para alterar um produto existente
```

```
app.put('/produtos/:id', (req, res) => {
    const id = parseInt(req.params.id);
    const produtoIndex = lista_produtos.produtos.findIndex(prod =>
prod.id === id);

    Verifique se o produto existe
    if (produtoIndex === -1) {
        return res.status(404).json({ mensagem: 'Produto não
encontrado' });
    }

    Defina a variável "descricao" com o valor da requisição
    const { descricao, valor, marca } = req.body;

    Atualize os dados do produto
    lista_produtos.produtos[produtoIndex].descricao = descricao;
    lista_produtos.produtos[produtoIndex].valor = valor;
    lista_produtos.produtos[produtoIndex].marca = marca;

    Atualize o hash para refletir a modificação do produto
    const hash =
crypto.createHash('sha1').update(JSON.stringify(lista_produtos)).digest
('hex');

    //apicache.clear('/produtos');

    Defina o cabeçalho ETag com o novo hash
    res.setHeader('ETag', hash)

    Envie uma resposta com os detalhes do produto atualizado
    res.status(200).json({
        message: 'Produto atualizado com sucesso!',
        product: lista_produtos.produtos[produtoIndex],
    });
});

Rota para excluir um produto
app.delete('/produtos/:id', (req, res) => {
    const id = parseInt(req.params.id);
    const produtoIndex = lista_produtos.produtos.findIndex(prod =>
prod.id === id);

    Verifique se o produto existe
```

```
    if (produtoIndex === -1) {
        return res.status(404).json({ mensagem: 'Produto não
encontrado' });
    }

    Remova o produto do array de produtos usando o índice correto
    lista_produtos.produtos.splice(produtoIndex, 1);

    Atualize o hash para refletir a exclusão do produto
    const hash =
crypto.createHash('sha1').update(JSON.stringify(lista_produtos)).digest
('hex');

    //apicache.clear('/produtos');

    Defina o cabeçalho ETag com o novo hash
    res.setHeader('ETag', hash);

    Envie uma resposta com uma mensagem de sucesso
    res.status(204).send();
});

Iniciando o servidor na porta 3000
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`Servidor rodando na porta ${PORT}`);
});
```