FIAP

FIAP GRADUAÇÃO

FIAP

SISTEMAS DE INFORMAÇÃO

MICROSERVICE AND WEB ENGINEERING

PROF. Thiago Xavier

Microsserviço - Status Code na prática

Agenda

Aula de Hoje:

- Status code na prática

Status code 401

- O status code 401, "Unauthorized" (Não Autorizado), é um código de resposta HTTP que indica que a solicitação requer autenticação do usuário

```
// Obtendo o token do header Authorization
//401 Unauthorized
router.get('/private', (req, res) => {
    // Obtendo o token do header Authorization
    const token = req.headers['authorization'];

    // Verificando se o token foi enviado e se é válido
    if (!token || token !== 'meuTokenSecreto') {
        // Retornando o status 401 Unauthorized se o token estiver ausente ou inválido
        return res.status(401).send('Unauthorized: token inválido');
    }

    // Se o token for válido, continuar com a requisição
    res.send('Area privada permitida!').sendStatus(200);
});
```

Status code 403

- O status code 403, "Forbidden" (Proibido), é um código de resposta HTTP que indica que o servidor entendeu a solicitação, mas se recusa a autorizá-la.

```
37  router.get('/admin', (req, res) => {
38      // Obtendo o token do header Authorization
39      const token = req.headers['authorization'];
40
41      // Verificando se o token foi enviado
42      if (!token) {
43          return res.status(401).send('Unauthorized: token não fornecido');
44      }
45
46      const user = tokensDatabase[token];
47      if (!user) {
48          return res.status(401).send('Unauthorized: token inválido');
49      }
50
51      // Verificando se o usuário tem acesso ao recurso administrativo
52      if (user.role !== 'admin') {
53          return res.status(403).send('Forbidden: você não tem permissão para acessar esta área');
54      }
55
56      res.send('Welcome to the admin area!');
57  });
```

Status 400

- O status code 400, "Bad Request" (Solicitação Inválida), é um código de resposta HTTP que indica que o servidor não pode ou não irá processar a solicitação devido a um erro no cliente.

```
60  //Exemplo bad request 400
61  router.post('/submit', (req, res) => {
62      const { username, email } = req.body;
63
64      // Verificando se os campos obrigatórios estão presentes
65      if (!username || !email) {
66          return res.status(400).send('Favor preencher todos os campos: username e email.');
67      }
68
69      // Simulando verificação de email válido
70      const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
71      if (!emailRegex.test(email)) {
72          return res.status(400).send('Bad Request: Invalid email format');
73      }
74
75      // Se tudo estiver certo, continuar com o processamento
76      res.send('Data submitted successfully').status(202);
77  });
78
```

Status 429

- O Middleware para rate limiting;
- Definição: Middleware é uma função em um aplicativo Express (ou outros frameworks)

```
79  // Objeto para rastrear as requisições de cada IP
80  const requestCounts = {};
81  const RATE_LIMIT = 5; // Limite de requisições permitido
82  const TIME_WINDOW = 60 * 1000; // Tempo da janela em milissegundos (1 minuto)
83
84  // Middleware para rate limiting
85  //Definição: Middleware é uma função em um aplicativo Express (ou outros frameworks)
86  //que processa requisições antes que elas cheguem à rota final ou resposta.
87
88  router.use((req, res, next) => {
89      const ip = req.ip;
90      //console.log(ip)
91      if (!requestCounts[ip]) {
92          requestCounts[ip] = { count: 1, firstRequest: Date.now() };
93      } else {
94          requestCounts[ip].count++;
95      }
96
97      const currentTime = Date.now();
98      const timePassed = currentTime - requestCounts[ip].firstRequest;
99
100     if (timePassed < TIME_WINDOW && requestCounts[ip].count > RATE_LIMIT) {
101         return res.status(429).send('Too Many Requests: Please try again later.');
102     }
103
104     if (timePassed >= TIME_WINDOW) {
105         requestCounts[ip].count = 1;
106         requestCounts[ip].firstRequest = Date.now();
107     }
108
109     next();
110 });
111
```

Status 429

- O status code 429, "Too Many Requests" (Muitas Solicitações), é um código de resposta HTTP que indica que o cliente enviou muitas solicitações em um determinado período de tempo.

```
112  // Rota de exemplo com rate limiting
113  router.get('/data', (req, res) => {
114      res.send('Aqui estão seus dados!');
115  })
```

Status code 404

- O status code 404, "Not Found" (Não Encontrado), é um código de resposta HTTP que indica que o servidor não encontrou o recurso solicitado. .

```
119
120  let items = [
121      { id: 0, name: 'item1' },
122      { id: 1, name: 'item2' },
123      { id: 2, name: 'item3' }
124  ];
125
126  router.get('/items/:id', (req, res) => {
127      const itemId = parseInt(req.params.id, 10);
128
129      // Verificando se o itemId é um número válido
130      if (isNaN(itemId)) {
131          return res.status(400).send('Bad Request: Invalid ID format');
132      }
133      const item = items.find(item => item.id === itemId);
134
135      if (item) {
136          res.json(item);
137      } else {
138          // Se o item não existir, retornando 404 Not Found
139          res.status(404).send('Item not found');
140      }
141  });
```

Status code 204 - exemplo put

- O status code 204, "No Content" (Sem Conteúdo), é um código de resposta HTTP que indica que a solicitação foi bem-sucedida, mas o servidor não tem nenhuma informação a enviar no corpo da resposta.

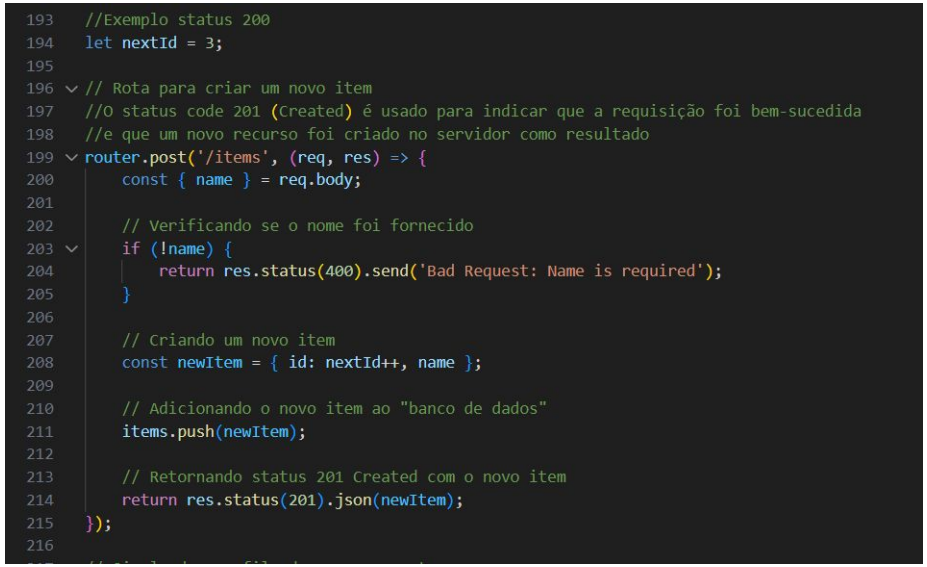
```
143  //O status code 204 (No Content) é usado para indicar que a requisição foi processada com sucesso,
144  //mas não há conteúdo a ser retornado na resposta
145  router.put('/items/:id', (req, res) => {
146      const itemId = parseInt(req.params.id, 10);
147      const newName = req.body.name;
148
149      // Verificando se o itemId é um número válido
150      if (isNaN(itemId)) {
151          return res.status(400).send('Bad Request: Invalid ID format');
152      }
153
154      // Buscando o índice do item no array
155      const itemIndex = items.findIndex(item => item.id === itemId);
156
157      // Verificando se o item existe
158      if (itemIndex !== -1) {
159          // Atualizando o item
160          items[itemIndex].name = newName;
161          // Retornando status 204 No Content
162          return res.status(204).send();
163      } else {
164          // Se o item não existir, retornando 404 Not Found
165          return res.status(404).send('Item not found');
166      }
167  });
```

Status code 204 - exemplo delete

```
169  //outro exemplo de 204 - no content
170  router.delete('/items/:id', (req, res) => {
171      const itemId = parseInt(req.params.id, 10);
172
173      // Verificando se o itemId é um número válido
174      if (isNaN(itemId)) {
175          return res.status(400).send('Bad Request: Invalid ID format');
176      }
177
178      // Buscando o índice do item no array
179      const itemIndex = items.findIndex(item => item.id === itemId);
180
181      // Verificando se o item existe
182      if (itemIndex !== -1) {
183          // Removendo o item
184          items.splice(itemIndex, 1);
185          // Retornando status 204 No Content
186          return res.status(204).send();
187      } else {
188          // Se o item não existir, retornando 404 Not Found
189          return res.status(404).send('Item not found');
190      }
191  });
192
```

Status 201 - Created

- O status code 201, "Created" (Criado), é um código de resposta HTTP que indica que a solicitação foi bem-sucedida e que um novo recurso foi criado como resultado da solicitação.



```
//Exemplo status 200
let nextId = 3;

// Rota para criar um novo item
//O status code 201 (Created) é usado para indicar que a requisição foi bem-sucedida
//e que um novo recurso foi criado no servidor como resultado
router.post('/items', (req, res) => {
    const { name } = req.body;

    // Verificando se o nome foi fornecido
    if (!name) {
        return res.status(400).send('Bad Request: Name is required');
    }

    // Criando um novo item
    const newItem = { id: nextId++, name };

    // Adicionando o novo item ao "banco de dados"
    items.push(newItem);

    // Retornando status 201 Created com o novo item
    return res.status(201).json(newItem);
});
```

Status 202 - Accepted

- O status code 202, "Accepted" (Aceito), é um código de resposta HTTP que indica que a solicitação foi recebida e compreendida pelo servidor, mas ainda não foi **processada completamente**.

```
// Simulando uma fila de processamento
const processingQueue = [];

// Rota para aceitar uma nova requisição de processamento
router.post('/envio-whatsapp', (req, res) => {
    const { data } = req.body;

    // Verificando se os dados foram fornecidos
    if (!data) {
        return res.status(400).send('Bad Request: Data is required');
    }

    // Simulando a aceitação da requisição para processamento futuro
    const requestId = Date.now();
    processingQueue.push({ requestId, data });

    // Retornando status 202 Accepted com o ID da requisição
    return res.status(202).json({ requestId, message: 'Request accepted for processing' });
});
```

Status 202 - Accepted

- Como não há uma resposta na primeira requisição, temos que oferecer de alguma forma o status. Vamos mostrar um exemplo.

```
237  // Rota para obter o status de uma requisição de processamento
238  router.get('/envio-whatsapp/:requestId', (req, res) => {
239      const requestId = parseInt(req.params.requestId, 10);
240
241      // Verificando se o requestId é um número válido
242      if (isNaN(requestId)) {
243          return res.status(400).send('Bad Request: Invalid request ID format');
244      }
245
246      // Buscando a requisição na fila de processamento
247      const request = processingQueue.find(req => req.requestId === requestId);
248
249      if (request) {
250          // Se a requisição existir, retornar o status 200 OK com os dados da requisição
251          return res.status(200).json(request);
252      } else {
253          // Se a requisição não existir, retornar o status 404 Not Found
254          return res.status(404).send('Request not found');
255      }
256  });
257
```

Dúvidas

Microsserviço - Vamos praticar

Exercício 1: Status 200 (OK)

Descrição: Crie uma rota GET que retorne informações de um usuário. Se o usuário for encontrado, retorne o status 200 e os detalhes do usuário.

Requisitos:

1. Crie uma rota `GET /users/:id` que retorne detalhes do usuário com o ID fornecido.
2. Retorne um status 200 (OK) com as informações do usuário em JSON.

Exercício 2: Status 201 (Created)

Descrição: Crie uma rota POST para cadastrar um novo usuário. Quando o usuário for criado com sucesso, retorne o status 201 e as informações do usuário.

Requisitos:

1. Crie uma rota `POST /users` que aceite um JSON com campos `name` e `email`.
2. Adicione o novo usuário a um array e retorne um status 201 (Created) com as informações do usuário.

Microsserviço - Vamos praticar

Exercício 3: Status 202 (Accepted)

Descrição: Crie uma rota POST que simule a criação de um processo assíncrono para enviar um e-mail. Retorne o status 202 com um ID para rastrear o envio.

Requisitos:

1. Crie uma rota `POST /email` que aceite um JSON com um campo `email`.
2. Retorne um status 202 (Accepted) com um `processId` para rastrear o envio do e-mail.

Exercício 4: Status 204 (No Content)

Descrição: Crie uma rota DELETE que exclua um usuário. Quando o usuário for excluído com sucesso, retorne o status 204.

Requisitos:

1. Crie uma rota `DELETE /users/:id` para excluir um usuário pelo ID.
2. Retorne um status 204 (No Content) após a exclusão.

Exercício 5: Status 400 (Bad Request)

Descrição: Crie uma rota POST que exige um campo obrigatório no corpo da requisição. Se o campo não estiver presente, retorne o status 400.

Requisitos:

1. Crie uma rota `POST /update-email` que aceite um JSON com um campo `email`.
2. Se o campo `email` não estiver presente, retorne um status 400 (Bad Request) com uma mensagem de erro.

Microsserviço - Vamos praticar

Exercício 6: Status 404 (Not Found)

Descrição: Crie uma rota GET para recuperar um usuário pelo ID. Se o usuário não for encontrado, retorne o status 404.

Requisitos:

1. Crie uma rota GET /users/:id para recuperar um usuário por ID.
2. Se o usuário não existir, retorne um status 404 (Not Found).