# Entendendo e Documentando RESTful APIs

# Gerindo Erros

# Gerindo Erros

---

“Naturalmente, quando fazemos **requisições RESTful**, receberemos como **retorno** um possível **erro**, seja por falha no **formato da requisição**, seja por **causas internas referentes ao servidor**.”

# Gerindo Erros

---

“Isso não significa que o **retorno** apresentado seja uma **mensagem clara**, que não deixa **dúvidas sobre o que aconteceu** de fato.”

# Gerindo Erros

---

“Pois bem, o intuito da **gerência de erros** em APIs Restful é **informar** ao **requisitante** uma **mensagem que retrate o que de fato ocorreu**. Mais do que isso, um **status code** que não seja genérico e sim, útil.”

# Gerindo Erros

---

Observe o erro 500 Internal Server Error.

```
jacksonpires:~/workspace $ curl -i -X POST https://jsonplaceholder.typicode.com/post
s -H "Content-Type: application/json" -d '{"title": "Hello!"'
HTTP/1.1 500 Internal Server Error
Date: Thu, 22 Sep 2016 12:19:00 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: __cfduid=d4447749775ba5dee1c244f9baecaac471474546740; expires=Fri, 22-Se
p-17 12:19:00 GMT; path=/; domain=.typicode.com; HttpOnly
X-Powered-By: Express
Vary: Origin, Accept-Encoding
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Pragma: no-cache
Expires: -1
Via: 1.1 vegur
Server: cloudflare-nginx
CF-RAY: 2e65b3e52f2e5504-ORD

SyntaxError: Unexpected end of JSON input
at Object.parse (native)
at parse (/app/node_modules/body-parser/lib/types/json.js:88:17)
at /app/node_modules/body-parser/lib/read.js:116:18
at invokeCallback (/app/node_modules/raw-body/index.js:262:16)
at done (/app/node_modules/raw-body/index.js:251:7)
at IncomingMessage.onEnd (/app/node_modules/raw-body/index.js:307:7)
at emitNone (events.js:86:13)
at IncomingMessage.emit (events.js:185:7)
at endReadableNT (_stream_readable.js:974:12)
at _combinedTickCallback (internal/process/next_tick.js:74:11)jacksonpires:~/workspa
ce $
```

# Gerindo Erros

---

“Apesar da stack informada no final (o que permite sabermos o que ocorreu), para o usuário final normalmente aparecerá apenas o status code, o que não é suficiente para indicar a origem do erro.”

# Classes dos HTTP Status Code

# Classes dos HTTP Status Code

---

“Existem 5 classes de HTTP Status Code. São elas:”

**1xx Informacional**: Códigos começados com **1** são conhecidos como códigos informacionais. A maioria deles não são usados nos dias atuais.

# Classes dos HTTP Status Code

---

**2xx Success**: Esses códigos indicam que ouve sucesso no intercâmbio entre o servidor e o cliente.

**3xx Redirection**: Os códigos **3xx** indicam que cliente deve fazer uma ação adicional antes da requisição estar completa.

# Classes dos HTTP Status Code

---

**4xx Client Error**: Nesse caso, o código indica que existe algo errado com a requisição do cliente.

**5xx Server Error**: O cliente enviou uma requisição válida, mas o servidor não foi capaz de processá-la com sucesso.

# Gerindo Erros

---

Veja detalhes de cada um dos possíveis Status Code

https://httpstatuses.com/