Desenvolvimento Backend

Aula 03 - Criando primeiro servidor web

Material Didático do Instituto Metrópole Digital - IMD

Apresentação

Nesta aula você aprenderá um pouco mais sobre o protocolo http, entendendo seu modo de funcionamento e mensagens envolvidas. Em seguida, verá como criar um servidor web utilizando a tecnologia Node.js, entendendo como tratar os métodos HTTP e os *paths* das requisições.

Objetivos

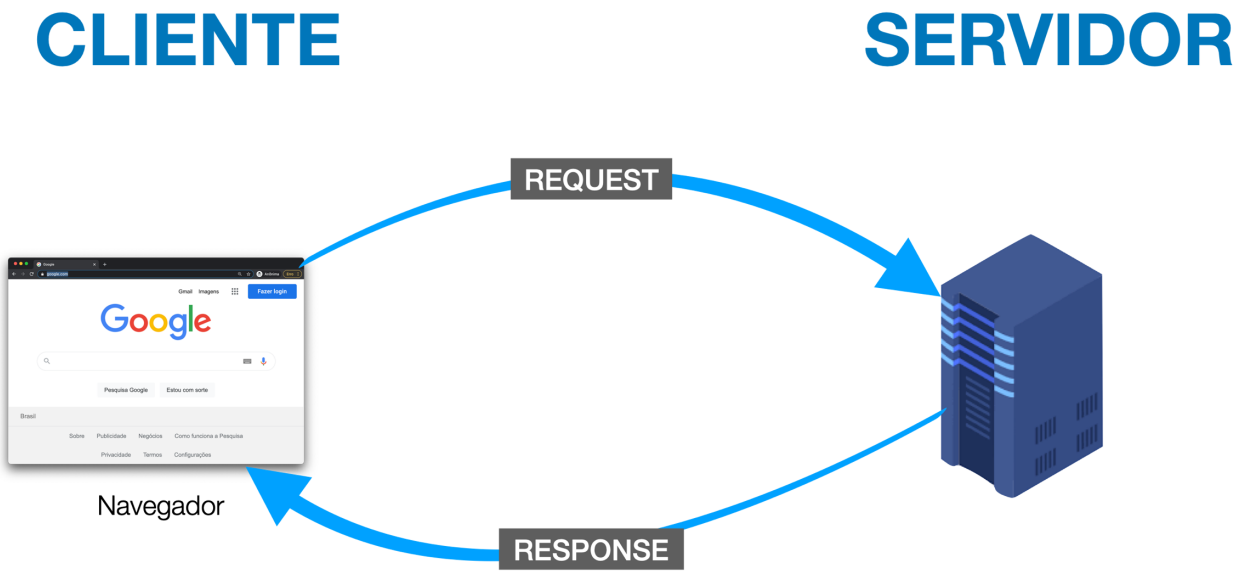
- Conhecer o protocolo HTTP e suas mensagens
- Conhecer como criar um servidor web com Node.js
- Entendercomo tratar o path da requisição e o método http da requisição

Como funciona um servidor HTTP

Link do video da aula: https://youtu.be/pCxUVrrUhPA

O protocolo HTTP é um importante protocolo utilizado na internet para realizar comunicação entre dois programas. É um protocolo cliente-servidor, ou seja, cada ator da comunicação realiza um papel diferente.

Figura 1 - Arquitetura cliente-servidor



Entender a diferença desses dois papéis na comunicação é importante para compreender melhor o protocolo. Dito isso, o cliente pode ser entendido como a parte **que inicia ativamente a comunicação**. Já o servidor é a parte passiva que aguarda por conexões vindas dos clientes.

No caso do HTTP, a comunicação ocorre através de pares de mensagens chamadas de *request* e *response*. O cliente é o responsável por fazer solicitações (*request*), enquanto o servidor responde (*response*) cada solicitação realizada pelo cliente.

Ao montar uma requisição, o cliente deve indicar explicitamente o endereço do recurso que deseja interagir. Esse endereçamento é feito através de uma URL, detalhada a seguir.

URL

A URL (Uniform Resource Locator) é um padrão estabelecido que permite indicar o endereço de algum recurso no servidor. A URL possui o seguinte formato:

Figura 2 - Formato da URL

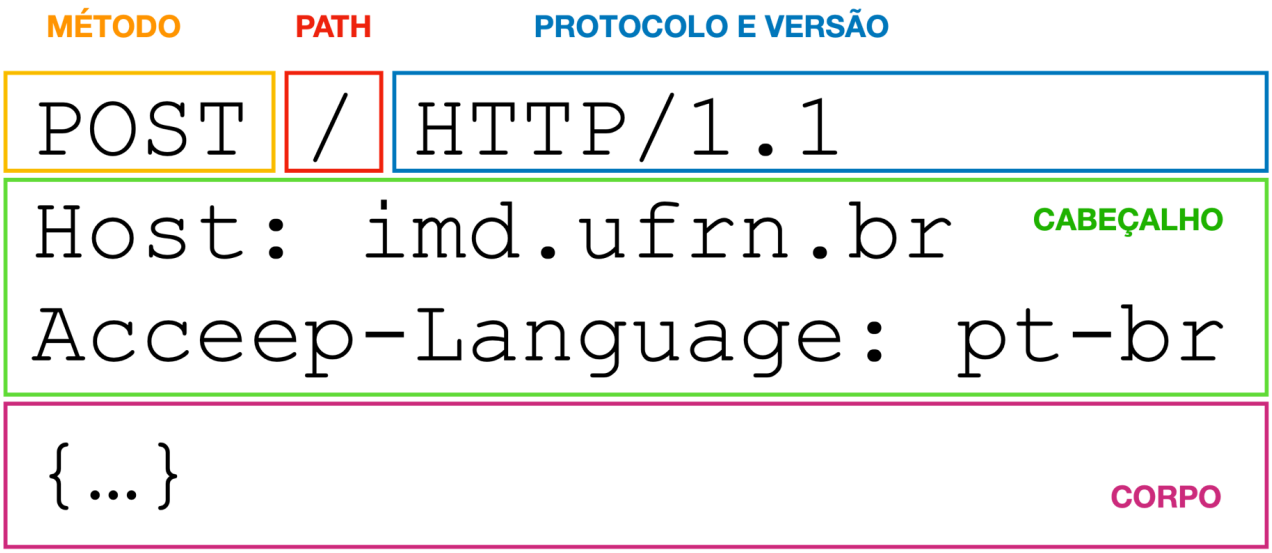
A primeira parte da URL é conhecida como *scheme* e indica o protocolo a ser utilizado. Em seguida, é indicado o endereço do host (servidor). Após o host, é colocada a porta da comunicação indicada após o sinal de dois-pontos (:). Depois, tem-se o *path*, que indica o caminho para o recurso. Por fim, é possível indicar um conjunto de parâmetros ao servidor (após o caractere de "?"), sendo cada parâmetro separado pelo caractere "&".

Além da URL, outras informações são importantes para montar toda requisição HTTP. As seções seguintes detalham os dados necessários para as requisições e respostas.

Request

O *request* é a requisição HTTP enviada pelo cliente ao servidor e que deve conter um conjunto de informações, conforme detalhamento abaixo:

Figura 3 - Quadro HTTP do request



- **Método** - método HTTP desejado;
- ***Path*** - caminho do recurso;
- **Protocolo e versão** - versão do protocolo HTTP a ser utilizado na comunicação;
- **Cabeçalho**: Informações de controle enviada na requisição;
- **Corpo**: dDados que serão enviados (carga útil).

Métodos HTTP

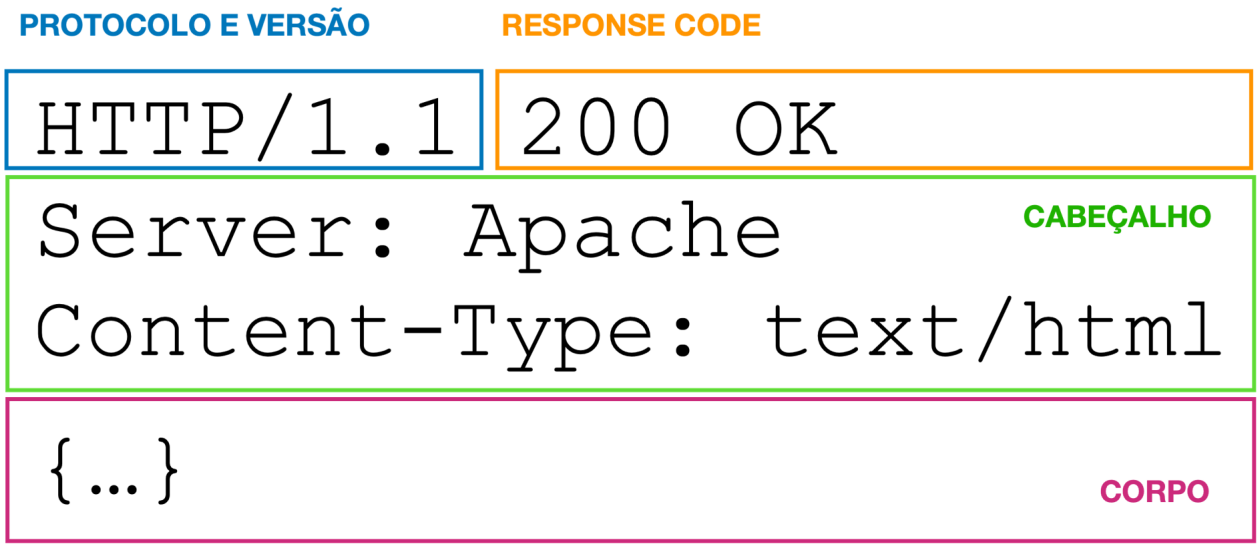
Vários métodos podem ser utilizados na requisição HTTP. Cada método indica uma função a ser executada no servidor. Veja as opções:

- **GET** - requisita algum recurso específico;
- **HEAD** - idêntico ao GET, mas não retorna o corpo da resposta;
- **POST** - envia uma entidade a um recurso específico;
- **PUT** - substitui a entidade pelo corpo enviado;
- **PATCH** - aplica modificações parciais a um recurso;
- **DELETE** - remove um recurso específico.

Response

O *response* é uma resposta a uma requisição enviada pelo servidor ao cliente e deve seguir um formato bem definido, conforme imagem abaixo.

Figura 4 - Quadro HTTP do response



- **Protocolo e versão** : versão do protocolo HTTP a ser utilizado na comunicação;
- ***Response code*** : código de retorno que indica *status* da resposta;
- **Cabeçalho**: informações de controle enviada na requisição;
- **Corpo**: dados que serão enviados como resposta (carga útil).

Response code

Na resposta de uma requisição HTTP, há um campo importante chamado *response code* que indica o status da operação. Acompanhe os valores possíveis para esse campo:

- **1XX** - resposta informativa indicada ao cliente;
- **2XX** - indica sucesso;
- **3XX** - indica ao cliente para redirecionar sua chamada ;
- **4xx** - indica erro na requisição por parte do cliente;
- **5xx** - indica erro no servidor ao tratar a requisição.

Servidor *web* com Node.js (introdução)

Link do video da aula: https://youtu.be/Xg38eApS7fY

Nesta aula, iremos criar um primeiro servidor web com Node.js. Para isso, acompanhe a videoaula e o passo a passo abaixo documentado.

Vamos iniciar o nosso projeto. Na pasta que deseja criar o projeto, execute o comando de inicialização do npm:

```
$ npm init
```

Em seguida, será criado um arquivo package.json. Abra esse arquivo e crie o script de *start*, conforme exemplo abaixo:

```
{
  "name": "aula03",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "author": "Gustavo",
  "license": "ISC",
}
```

Agora crie o arquivo Javascript index.js que será o ponto de entrada do programa com o seguinte conteúdo:

```
const http = require('http')

const server = http.createServer((req, res) => {
    console.log(req.url)
        res.writeHead(200, { 'Content-Type': "text/json" })
        res.write(JSON.stringify({ msg: "Hello World!!" }))
        res.end()
})

server.listen(8080, () => {
    console.log('Servidor pronto na porta 8080!')
})
```

Nesse caso, o programa cria um servidor web na porta 8080 que responde qualquer requisição feita com sucesso. A biblioteca utilizada (http) é do próprio Node.js e por isso não é necessário instalar nenhum pacote adicional.

Para executar o programa, digite:

```
$ npm start
```

Adicionando nodemon

Como vimos anteriormente, o nodemon aumenta a produtividade do desenvolvimento, visto que recarrega o sistema sempre que uma mudança no código é feita. Para adicioná-lo ao projeto, execute os seguintes passos:

```
$ npm install --save-dev nodemon
```

Em seguida, edite o arquivo package.json para utilizar o nodemon:

```
{
  "name": "aula03",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "nodemon --exec node index.js"
  },
  "author": "Gustavo",
  "license": "ISC",
}
```

Servidor *web* com Node.js - Path

Link do video da aula: https://youtu.be/t5-eEN-yqPo

Vamos incrementar mais o nosso servidor web realizando um tratamento melhor da requisição através da interpretação do *path* da requisição.

A informação do *path* pode ser obtida pelo parâmetro "req" da requisição através da propriedade "req.url". Logo, é possível examinar o conteúdo dessa variável e criar diferentes tratamentos. Acompanhe na videoaula a explicação detalhada do exemplo abaixo:

```
const http = require('http')
const { v4: uuidv4 } = require('uuid')

const server = http.createServer((req, res) => {
  switch (req.url) {
    case "/aluno":
      res.writeHead(200, { 'Content-Type': 'text/json' })
      res.write(JSON.stringify({ "msg": "Aluno criado", "path":
```

```
req.url }))
      res.end()
      break
    default:
      res.writeHead(404, { 'Content-Type': 'text/json' })
      res.write(JSON.stringify({ "msg": "Path não encontrado",
"path": req.url }))
      res.end()
  }
})

server.listen(8080, () => {
  console.log('Servidor iniciado na porta 8080')
})
```

Servidor *web* com Node.js - Métodos HTTP

Link do video da aula: https://youtu.be/ur06moc3Q4I

Nesta aula, vamos continuar o desenvolvimento do servidor web através da análise do método HTTP. O método da requisição http pode ser acessado através da propriedade "req.method". Com isso, é possível criar diferentes tratamentos a depender do método.

Acompanhe a explicação detalhada na videoaula do exemplo do código abaixo:

```
const http = require('http')

const server = http.createServer((req, res) => {
    console.log(req.url)

    switch (req.url) {
        case '/aluno':
            alunoRoute(req, res)
            break;
        default:
            res.writeHead(404, { 'Content-Type': "text/json" })
            res.write(JSON.stringify({ msg: "Path não encontrado"
}))
            res.end()
    }
```

```
})

server.listen(8080, () => {
    console.log('Servidor pronto na porta 8080!')
})

function alunoRoute(req, res) {
    switch (req.method){
        case 'GET':
            res.writeHead(200, { 'Content-Type': "text/json" })
            res.write(JSON.stringify({ alunos: ["Gustavo", "João"]
}))
            res.end()
            break;
        case 'POST':
            res.writeHead(200, { 'Content-Type': "text/json" })
            res.write(JSON.stringify({ msg: "Aluno criado" }))
            res.end()
            break
        default:
            res.writeHead(400, { 'Content-Type': "text/json" })
            res.write(JSON.stringify({ msg: "Operação não
suportada!" }))
            res.end()
            break
    }
}
```

Resumo

Nesta aula você conheceu o protocolo http e viu como criar um servidor web utilizando apenas o Node.js sem auxílio de nenhuma biblioteca externa. Viu como tratar a requisição fornecendo diferentes tratamentos através da interpretação do *path* e do método da requisição.