

Symfony



# SISTEMAS CLIENTE SERVIDOR

NodeJS, PHP, Symfony,  
Bootstrap

Ser humilde com os superiores é obrigação, com os colegas é cortesia, com os inferiores é nobreza.

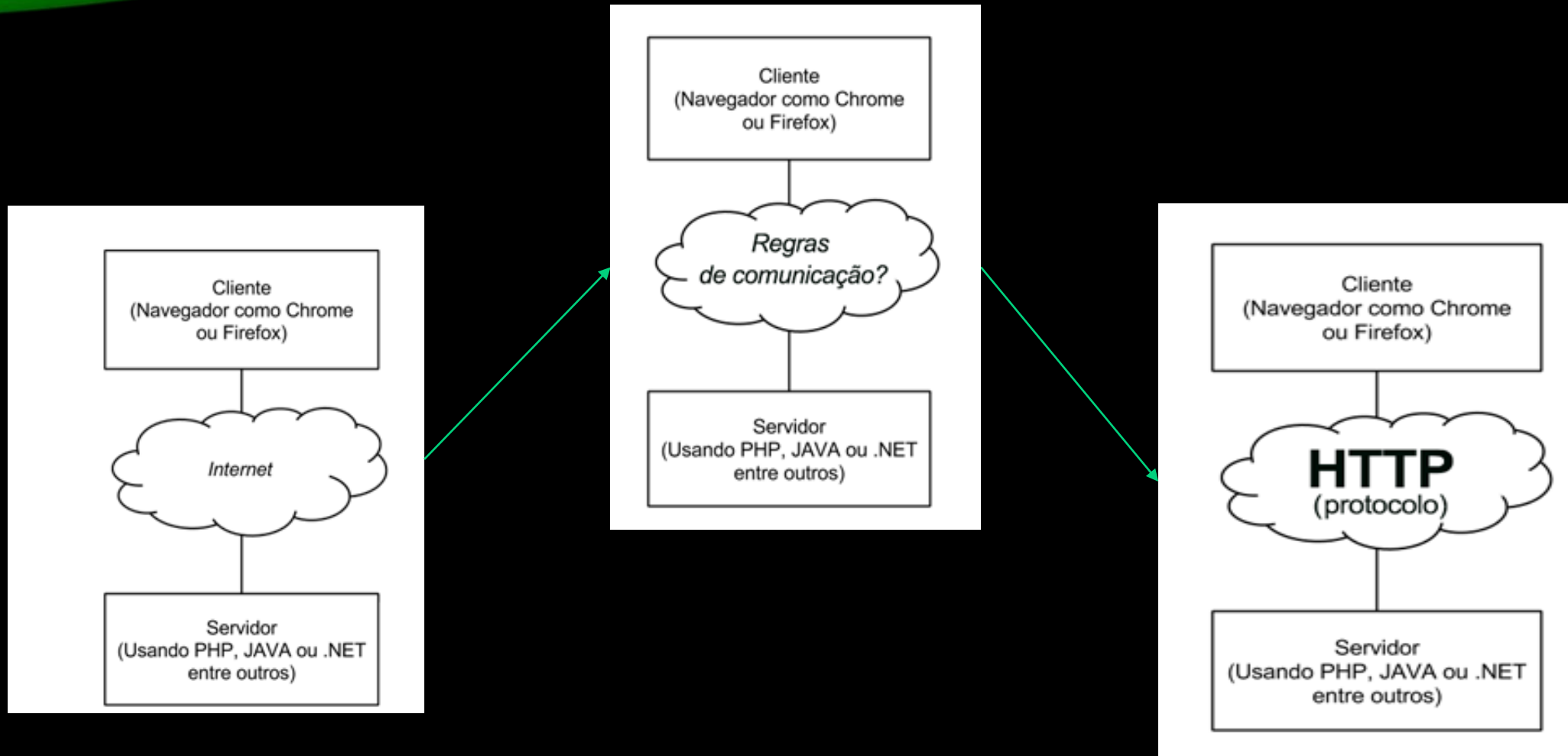
Benjamin Franklin

# ANTES DE INICIAR

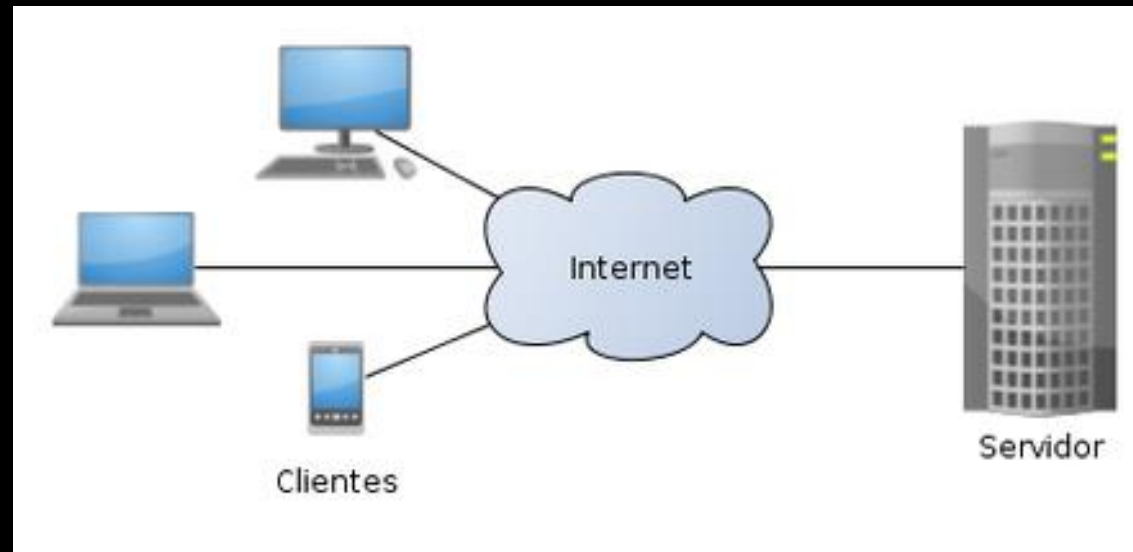
- Reforçar conhecimento em JavaScript e TypeScript
  - <https://www.w3schools.com/js/>

# HTTP *Hypertext Transfer Protocol*

## HTTPS *Hyper Text Transfer Protocol Secure*



# CLIENTE X REGRAS X SERVIDOR



Para comunicação na internet, sempre vai ser utilizado algum protocolo entre o cliente e o servidor. Sendo um dos mais importantes o HTTP, na comunicação as regras de como o processo vai ocorrer são definidas dentro do protocolo utilizado. Nas nossas aulas vamos utilizar o HTTP.

# VERBOS HTTP



**GET**

Recupera informações  
Listar/exibir



**POST**

Gravar/Inserir  
Login/Cadastros



**PUT**

Atualizar informações



**DELETE**

Excluir informações

# VERBOS HTTP

## GET

O método GET solicita a representação de um recurso específico. Requisições utilizando o método GET devem retornar apenas dados.

## HEAD

O método HEAD solicita uma resposta de forma idêntica ao método **GET**, porém sem conter o corpo da resposta.

## 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.

# VERBOS HTTP

## 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.

## CONNECT

O método CONNECT estabelece um túnel para o servidor identificado pelo recurso de destino.



# VERBOS HTTP

## OPTIONS

O método OPTIONS é usado para descrever as opções de comunicação com o recurso de destino.

## TRACE

O método TRACE executa um teste de chamada *loop-back* junto com o caminho para o recurso de destino.

## PATCH


O método PATCH é utilizado para aplicar modificações parciais em um recurso

# VERBOS HTTP

SWAPI  
The Star Wars API

<https://swapi.dev/>



GET  <https://swapi.co/api/people/1/>

Key	Value
New key	Value

Authorization Headers Body Pre-request Script Tests

Key	Value
New key	Value

Body Cookies (1) Headers (13) Test Results

**Allow** → GET, HEAD, OPTIONS

**CF-RAY** → 4a738be1dd2121da-EWR

**Connection** → keep-alive

**Content-Encoding** → gzip

**Content-Type** → application/json

**Date** → Mon, 11 Feb 2019 02:57:09 GMT

**Etag** → W/"145c70f4eca80b4752674d42e5bf1bcf"

**Expect-CT** → max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"

**Server** → cloudflare

**Transfer-Encoding** → chunked

# EXERCÍCIO

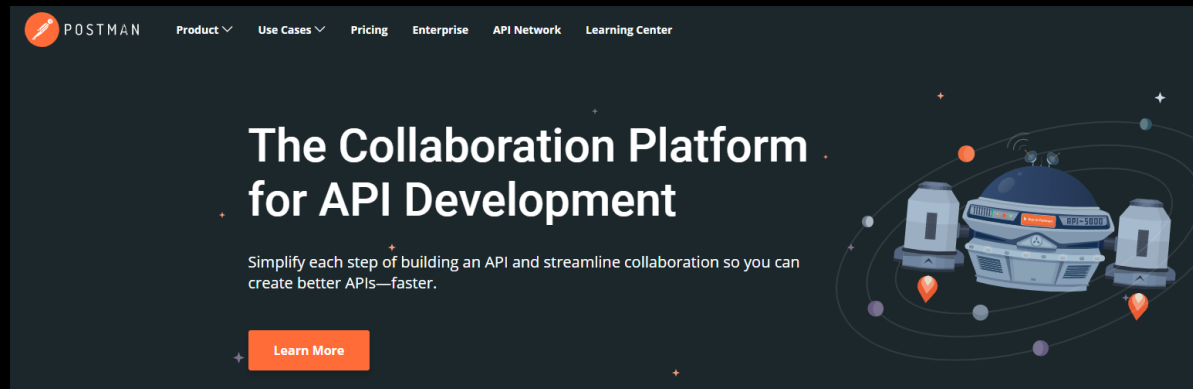
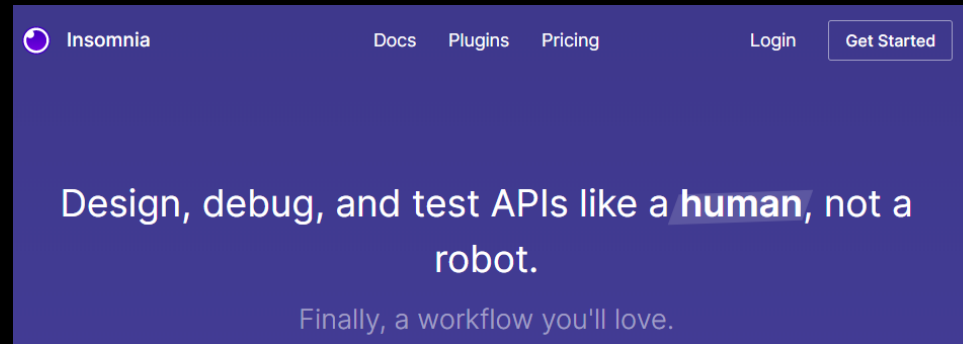
1 – Informe quais endpoints são necessários para se obter todos os atores de todos os filmes onde Obi-Wan Kenobi apareceu.

2 – Qual o ID e Altura do Obi-Wan Kenobi ?

# LEITURAS:

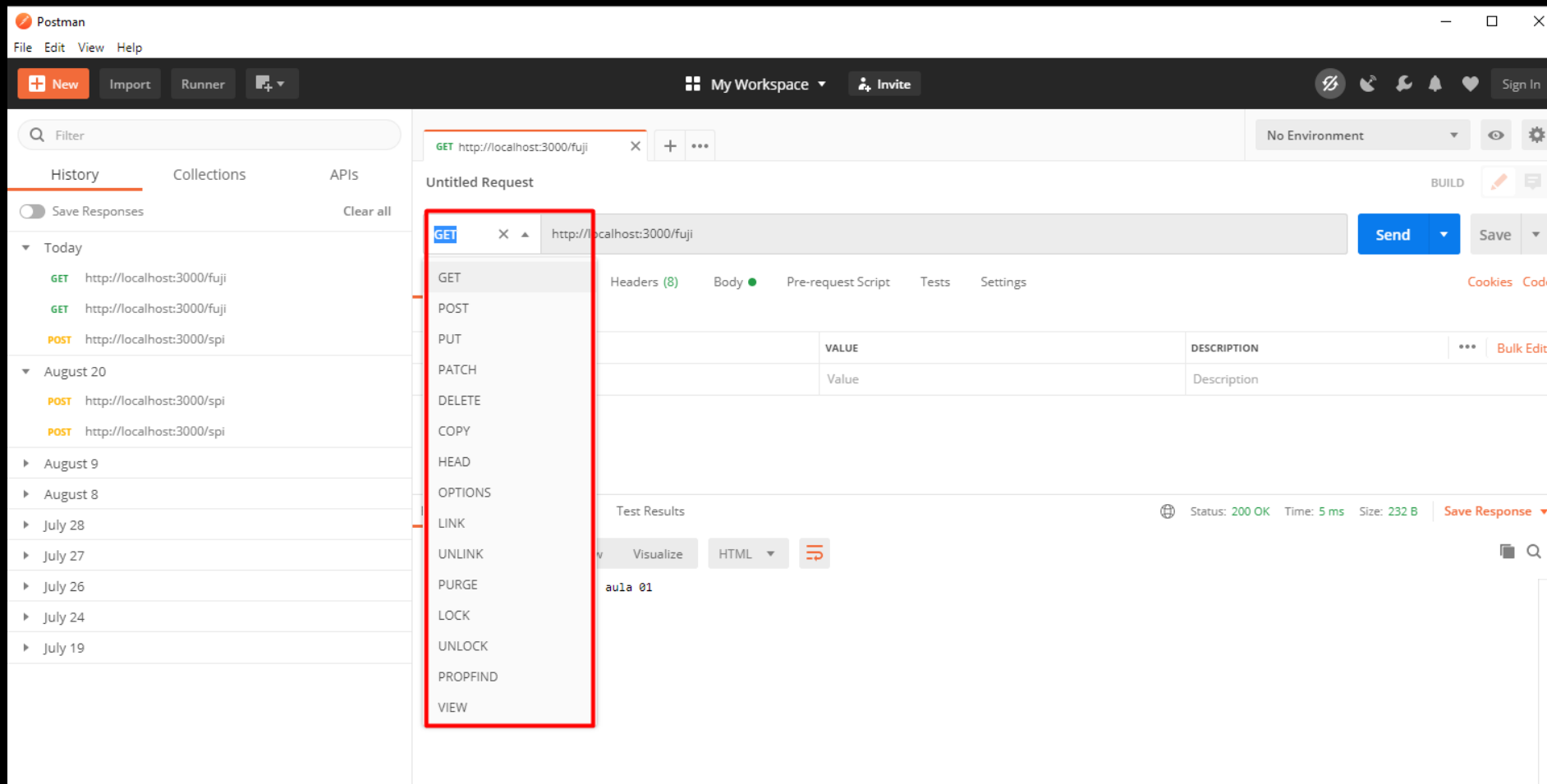
1. <https://www.devmedia.com.br/servicos-restful-verbos-http/37103>
2. <https://developer.mozilla.org/pt-BR/docs/Web/HTTP>
3. <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>
4. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

- <https://insomnia.rest/>



- <https://www.postman.com/>

NESSA MATERIAL, VAMOS UTILIZAR  
O POSTMAN.





# AS REQUISIÇÕES (RESPOSTAS)

- 2XX, 200, 203, 207 -> Respostas de Sucesso
- 3XX, 300, 301, 302 -> Mensagem de redirecionamento
- 4XX, 401, 404, 405 -> Respostas de erro do cliente
- 5XXX, 500, 502, 503 -> Respostas de erro do servidor



# SERVIÇOS WEB

## JSON (*JavaScript Object Notation*)

```
1  {  
2    "aula": {  
3      "professor": "Rodrigo Fujioka ",  
4      "disciplina": "Programação"  
5    }  
6  }
```

## XML (*Extensible Markup Language*)

```
<aula>  
  <professor>Rodrigo Fujioka </professor>  
  <disciplina>Programação</disciplina>  
</aula>
```

[https://www.w3schools.com/js/js\\_json\\_xml.asp](https://www.w3schools.com/js/js_json_xml.asp)

# API

**API** do acrônimo de **Application Program Interface** é um conjunto de rotinas que realiza comunicação entre aplicações que desejam compartilhar suas rotinas (SCHOOL OF NET, 2020).

API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web. A sigla API refere-se ao termo em inglês "**Application Programming Interface**" que significa em tradução para o português "Interface de Programação de Aplicativos" (CANALTECH, 2020).

Uma **API é criada quando uma empresa de software tem a intenção de que outros criadores de software** desenvolvam produtos associados ao seu serviço (CANALTECH, 2020).

# JSON

- **JavaScript Object Notation**  
(<https://www.json.org/json-pt.html>)
- [https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global\\_Objects/JSON](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/JSON)
- JSON é uma sintaxe para serialização de objetos, matrizes, números, strings, booleanos, e null. Baseia-se em sintaxe Javascript, mas é distinta desta: alguns Javascript não são JSON, e alguns JSON não são Javascript. (MONZILA, 2020)

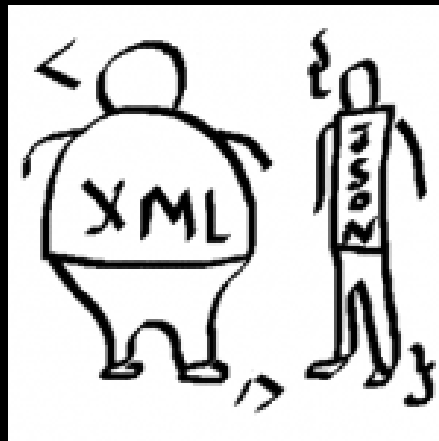


# JSON - JAVASCRIPT OBJECT NOTATION,

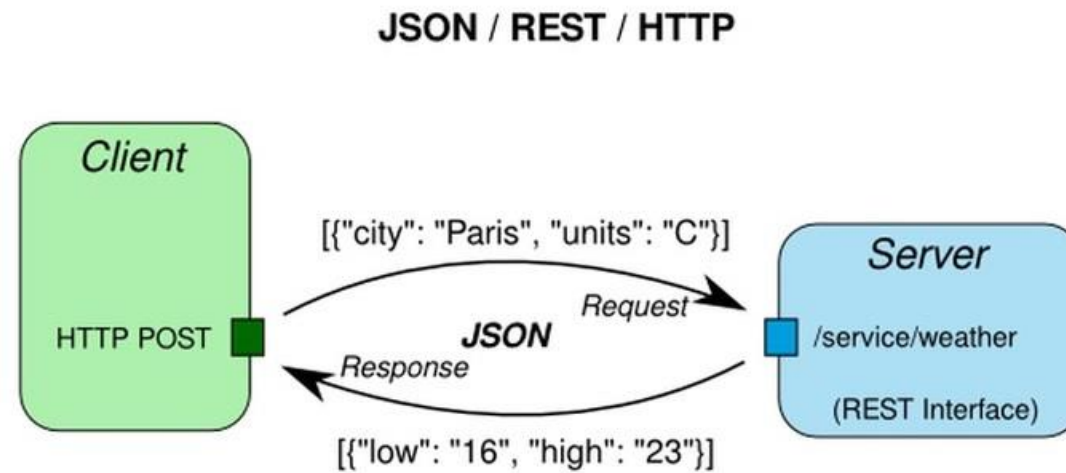
- “JSON é basicamente um formato **leve de troca** de informações/dados entre sistemas  
Leia mais em” ([Introdução: JSON http://www.devmedia.com.br/introducao-json/23166#ixzz45eKV5umO](http://www.devmedia.com.br/introducao-json/23166#ixzz45eKV5umO))

Leia

<http://www.infoq.com/br/news/2013/11/xml-json-performance>



<XML>  
{JSON}



Source: Safety Net



# JAVASCRIPT X JSON

## JavaScript e JSON diferenças

JavaScript tipo	JSON diferenças
Objetos e Arrays	Os nomes das propriedades devem ser strings com aspas duplas; as vírgulas à direita são proibidas.
Números	Zeros à esquerda são proibidos; um ponto decimal deve ser seguido por pelo menos um dígito.
Strings	Apenas um conjunto limitado de caracteres pode ser escapado; certos caracteres de controle são proibidos; o separador de linha Unicode (U+2028) e o separador de parágrafo (U+2029) caracteres são permitidos; strings devem ter aspas duplas.Veja o exemplo a seguir, onde <code>JSON.parse()</code> funciona bem e um <code>SyntaxError</code> é lançado ao avaliar o código como JavaScript: <code>var code = '"\u2028\u2029"; JSON.parse(code); // works fine eval(code); // fails</code>

# CRIE UMA CLASSE PHP

- Crie a classe Usuario
  - Id, nome idade;
- Crie uma classe que exiba o objeto convertido em JSON.

```
<?php

class Usuario implements JsonSerializable
{
    private $id;
    private $nome;
    private $idade;

    public function __construct(){}

    /**
     * @return mixed
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * @param mixed $id
     */
    public function setId($id)
    {
        $this->id = $id;
    }
}
```

# OBCJET TO JSON

```
1  <?php
2
3  require_once ("Usuario.php");
4
5  function exibeJson($usuario){
6
7      $usuarioJson = json_encode($usuario);
8      print($usuarioJson);
9  }
10
11  $usuario = new Usuario();
12  $usuario->setId(id: 1);
13  $usuario->setNome(nome: "Rodrigo da Cruz Fujioka");
14  $usuario->setIdade(idade: 37);
15
16  exibeJson($usuario);
17
18  ?>
```



# DESERIALIZANDO

- Exemplo PHP

```
1  <?php
2
3  require_once ("Usuario.php");
4
5      function exibeDados($usuario){
6          print("ID :" . $usuario->id);
7          print("<br/>NOME :" . $usuario->nome);
8          print("<br/>Idade:" . $usuario->idade);
9      }
10
11
12  $usuarioJSon = '{"id":1,"nome":"Rodrigo da Cruz Fujioka","idade":37}'
13
14  $usuario = json_decode($usuarioJSon);
15
16  exibeDados($usuario);
17
18  ?>
```

# EXECUTANDO

- Na pasta do Projeto
  - `php -S localhost:3000 -t .` // Vai iniciar um servidor embutido do PHP
  - Quem estiver utilizando o XAAMP pode apenas chamar no navegador o arquivo na porta configurada. `localhost/pasta/nomeArquivo.php`

# OBSERVE O JSON ABAIXO

```
1  {  
2  professor: {  
3      id: 123456789,  
4      nome: "Rodrigo da Cruz Fujioka",  
5      login: "fujioka",  
6      email: "rcf4@cin.ufpe.br"  
7  }  
8  }
```

# EXERCÍCIO 1

- Crie um arquivo na linguagem que você é mais fluente que represente o JSON professor.
- Crie uma classe que Transforme o objeto em PHP e um Json em PHP.

# EXERCÍCIO 2

- <https://developer.mozilla.org/pt-BR/docs/Aprender/JavaScript/0>
- Mapeie quais entidades são representadas no JSON
- Mapeie uma classe PHP ou JS para o JSON da Figura.

```

1  {
2      "squadName": "Super hero squad",
3      "homeTown": "Metro City",
4      "formed": 2016,
5      "secretBase": "Super tower",
6      "active": true,
7      "members": [
8          {
9              "name": "Molecule Man",
10             "age": 29,
11             "secretIdentity": "Dan Jukes",
12             "powers": [
13                 "Radiation resistance",
14                 "Turning tiny",
15                 "Radiation blast"
16             ]
17         },
18         {
19             "name": "Madame UpperCut",
20             "age": 39,
21             "secretIdentity": "Jane Wilson",
22             "powers": [
23                 "Million tonne punch",
24                 "Damage resistance",
25                 "Superhuman reflexes"
26             ]
27         },
28         {
29             "name": "Eternal Flame",
30             "age": 1000000,
31             "secretIdentity": "Unknown",
32             "powers": [
33                 "Immortality",
34                 "Heat Immunity",
35                 "Inferno",
36                 "Teleportation",
37                 "Interdimensional travel"
38             ]
39         }
40     ]
41 }

```

```
@lombok.Data
public class Squad {
    private String squadName;
    private String homeTown;
    private long formed;
    private String secretBase;
    private boolean active;
    private List<Member> members;
}
```

```
@lombok.Data
public class Member {
    private String name;
    private long age;
    private String secretIdentity;
    private List<String> powers;
}
```

```
export interface Squad {
    squadName: string;
    homeTown: string;
    formed: number;
    secretBase: string;
    active: boolean;
    members: Member[];
}
```

```
export interface Member {  
    name: string;  
    age: number;  
    secretIdentity: string;  
    powers: string[];  
}
```

```

1  {
2    "squadName": "Super hero squad",
3    "homeTown": "Metro City",
4    "formed": 2016,
5    "secretBase": "Super tower",
6    "active": true,
7    "members": [
8      {
9        "name": "Molecule Man",
10       "age": 29,
11       "secretIdentity": "Dan Jukes",
12       "powers": [
13         "Radiation resistance",
14         "Turning tiny",
15         "Radiation blast"
16       ]
17     },
18     {
19       "name": "Madame Uppercut",
20       "age": 39,
21       "secretIdentity": "Jane Wilson",
22       "powers": [
23         "Million tonne punch",
24         "Damage resistance",
25         "Superhuman reflexes"
26       ]
27     },
28     {
29       "name": "Eternal Flame",
30       "age": 1000000,
31       "secretIdentity": "Unknown",
32       "powers": [
33         "Immortality",
34         "Heat Immunity",
35         "Inferno",
36         "Teleportation",
37         "Interdimensional travel"
38       ]
39     }
40   ]
41 }

```





DÚVIDAS ?

- **Dúvidas sobre alguma das coisas que vimos ?**

# JAVASCRIPT

- Variáveis, dados e operações
- Funções
- Constantes
- Operadores Lógicos e Condicionais
- Laços de repetição
- TypeScript e Node.JS



# VAMOS VER ?



## JavaScript Demo: Expressions - Spread syntax

```
1 function sum(x, y, z) {
2   return x + y + z;
3 }
4
5 const numbers = [1, 2, 3];
6
7 console.log(sum(...numbers));
8 // expected output: 6
9
10 console.log(sum.apply(null, numbers));
11 // expected output: 6
12
```

(Monzila, 2020)

# VARIÁVEIS

- Não tem tipagem estática.
  - Você não precisa definir o tipo como em Java, C#, etc.
    - `Integer valor = 8; // Essa variável armazena apenas valores inteiros.`
- Tipagem dinâmica.

# VARIÁVEIS

```
variaveis.html - frameworkswb - Visual Studio Code

EXPLORER
> OPEN EDITORS
  FRAMEWORKSWEB
    > atendimentoapi
    > expressjs
    > javascript-intro
    <> variaveis.html
    > pdf
    > praticas
    .gitignore
    LICENSE
    README.md

variaveis.html
javascript-intro > <> variaveis.html > html > body > script
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <p id="aulafuji"></p>
5      <script>
6          var codigo = 202009101;
7          var nome = "Frameworks Web";
8          var opcional = false;
9          var semestre = 2020.2;
10         var cargaHoraria = 80;
11
12         var disciplinas = ['Frameworks Web', 'Express', 'Mobile'];
13
14         var disciplina = {
15             codigo : 202009101,
16             nome : "Frameworks Web",
17             opcional : false,
18             semestre : 2020.2,
19             cargaHoraria: 80,
20
21         };
22         console.log(disciplina);
23         console.log(disciplinas);
24         document.getElementById("aulafuji").innerHTML = nome;
25     </script>
26
27 </body>
28 </html>
29
```

variaveis.html

Arquivo | D:/ambiente/siste... ☆ (≡) C: R Outros favoritos

Apps Arquitetura Dev-1 Dev-2 pos

Frameworks Web

Elements Console Sources >> ⚙ ⋮ ✕

top 🔍 Filter Default levels ⚙

variaveis.html:22

```
{codigo: 202009101, nome: "Frameworks Web", opcional: false, semestre: 2020.2, cargaHoraria: 80}
```

cargaHoraria: 80  
codigo: 202009101  
nome: "Frameworks Web"  
opcional: false  
semestre: 2020.2  
▶ \_\_proto\_\_: Object

variaveis.html:23

```
(3) ["Frameworks Web", "Express", "Mobile"]
```

0: "Frameworks Web"  
1: "Express"  
2: "Mobile"  
length: 3  
▶ \_\_proto\_\_: Array(0)

>

# OPERADORES

javascript-intro > <> operacoes.html > html > body > script

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <p id="aulafuji"></p>
5          <script>
6              var anoNascimento = 1982;
7              var anoAtual = 2020;
8              var idade = anoAtual - anoNascimento;
9              var anoPassado = anoAtual-1;
10             var anoFuturo = anoAtual + 1;
11
12             //anoatual += 1;  2021
13             //anoatual -= 1;  2019;
14
15             console.log(anoFuturo);
16         </script>
17
18 </body>
19 </html>
```

# FUNÇÕES

- Escopo das variáveis;
- Chamada.

javascript-intro > <> funcoes.html > html > body > script >

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <p id="aulafuji"></p>
5      <script>
6          function somar(valorA, valorB){
7              var soma = valorA + valorB;
8              return soma;
9          }
10
11         function exibirNome(nome){
12             window.alert(nome);
13         }
14
15         var soma = somar(4,5);
16         console.log(soma);
17         exibirNome("Rodrigo Fujioka");
18
19     </script>
20
21 </body>
22 </html>
```

# CONDICIONAIS

- IF , ELSE

```
EXPLORER  ...  <> variaveis.html  <> funcoes.html  <> estruturacondicional.html X  <> operacoes.html  <> verificarAnoAtual.html

OPEN EDITORS
FRAMEWORKSWEB
> atendimentoapi
> expressjs
v javascript-intro
  <> estruturacondicio... U
  <> funcoes.html U
  <> operacoes.html U
  <> variaveis.html U
> pdf
> praticas
.gitignore
LICENSE
README.md

OUTLINE

javascript-intro > <> estruturacondicional.html > html > body > script > verificarAnoAtual.html
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <p id="aulafuji"></p>
5      <script>
6          function verificarMaioridade(idade){
7              if(idade >= 18 ){
8                  return "Maior de idade"
9              }else{
10                 return "Menor de idade"
11             }
12         }
13         function verificarAnoAtual(anoTeste){
14             if(anoTeste===2020){ //experimente ==
15                 console.log("É ano atual");
16             }else{
17                 console.log("Não é ano atual");
18             }
19         }
20
21         var retorno = verificarMaioridade(18);
22         console.log(retorno);
23         retorno = verificarMaioridade(16);
24         console.log(retorno);
25
26         verificarAnoAtual("2020");
27         verificarAnoAtual(2020);
28     </script>
29
30 </body>
31 </html>
```

# CONDICIONAIS

- Switch

The image shows a development environment with Visual Studio Code on the left and a web browser on the right. The VS Code editor displays a file named `estruturacondicionalswitch.html` with the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4   <p id="aulafuji"></p>
5   <script>
6     function verificarMarcaCerta(marca){
7
8       switch(marca) {
9         case 'Fujioka' : return "Marca Certa";
10        case 'xingling' : return "Outra Marca";
11        default: return "Marca desconhecida";
12      }
13
14    }
15
16
17    var retorno = verificarMarcaCerta("Fujioka");
18    console.log(retorno);
19    retorno = verificarMarcaCerta("xingling");
20    console.log(retorno);
21    retorno = verificarMarcaCerta("fuji");
22    console.log(retorno);
23  </script>
24
25 </body>
26 </html>
```

The web browser on the right shows the rendered page. The console is open, displaying the output of the JavaScript code:

Log Entry	Source
Marca Certa	<a href="#">estruturacondicionalswitch.html:18</a>
Outra Marca	<a href="#">estruturacondicionalswitch.html:20</a>
Marca desconhecida	<a href="#">estruturacondicionalswitch.html:22</a>



# OPERADORES

- && (E)
- || (OU)
- ! (NOT)

```
avascript-intro > <> estruturacondicionalswitch.html > ...
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <p id="aulafuji"></p>
5      <script>
6          function verificarMarcaCerta(marca){
7             
8              if(marca === 'Fujioka' && marca === 'Fuji'){
9                  return "Marca certa"
10             }
11             
12             if(marca !== 'xingling' ){
13                 return "Outra Marca"
14             }else {
15                 return "Marca XingLing"
16             }
17             
18         }
19         var retorno = verificarMarcaCerta("Fujioka");
20         console.log(retorno);
21         retorno = verificarMarcaCerta("xingling");
22         console.log(retorno);
23         retorno = verificarMarcaCerta("Fuji");
24         console.log(retorno);
25         retorno = verificarMarcaCerta("TainheilenGambielMeloOka");
26         console.log(retorno);
27     </script>
28 
29 </body>
30 </html>
```



# LAÇOS DE REPETIÇÃO

- setInterval(função, milisegundos);
- setTimeout(função, milisegundos);

```
javascript-intro > <> intervalor_timeout.html > html > body > script
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <p id="aulafuji"></p>
5      <script>
6
7          function exibeData(){
8              console.log(new Date())
9          }
10
11          //setInterval(exibeData, 1000);
12          setInterval(() => {
13              exibeData();
14          }, 1000);
15          //setTimeout(exibeData, 1000);
16          setTimeout(() => {
17              exibeData();
18          }, 5000);
19
20      </script>
21
22  </body>
23  </html>
```

# EXERCÍCIO

- 1 – Crie em javascript uma calculadora.
- 2 – Crie em uma função para exibir todos os números pares em um intervalo de números ex entre 1 e 100.
- 3 – Crie uma função que verifique o sexo e a idade do usuário.
  - A -> Se Sexo Masculino e Idade  $\geq 18$  pode entrar no Bar com taxa.
  - A2 -> Se Sexo Masculino e Idade  $< 18$  não pode entrar no Bar.
  - B -> Se sexo Feminino e Idade  $\geq 18$  pode entrar no bar sem taxa.
  - B2 -> Se sexo Feminino e Idade  $< 18$  Não pode entrar no bar.

# NODE.JS

- Node.js pode ser definido como um ambiente de execução Javascript server-side (Opus-software)
  - Realizar leitura do material complementar **AULA02\_Leitura01\_Node.pdf** e [https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Express\\_Nodejs/Introdu%C3%A7%C3%A3o](https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Express_Nodejs/Introdu%C3%A7%C3%A3o)

# EXPRESS

- Criado por TJ Holowaychuk. <https://github.com/expressjs/express>
- Framework web rápido, flexível e minimalista para Node.js (<https://expressjs.com/pt-br/>)
  - Quem usa: Fox Sports, PayPal, Uber e também pela própria IBM (Wikipedia)

MÃOS A OBRA.

- Crie um repositório vazio no github.
- Clone ele no seu ambiente local.
  - `git clone nomeRepositorio`



# CLONAR REPOSITÓRIO

The screenshot shows the GitHub interface for the repository 'rodrigofujioka / clienteservidor'. At the top, there are navigation tabs: Code (selected), Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the tabs, the repository name is displayed, followed by 'master' branch, '1 branch', and '0 tags'. A 'Go to file' button, an 'Add file' button, and a green 'Code' button are visible. The 'Code' button is open, showing options: 'Clone with HTTPS' (with a link to 'Use SSH'), 'Use Git or checkout with SVN using the web URL.' (with a text box containing 'https://github.com/rodrigofujioka/clienteservidor'), 'Open with GitHub Desktop', and 'Download ZIP'. Below the file list, the README.md content is partially visible, showing the title 'clienteservidor'.

rodrigofujioka / clienteservidor

<> Code   ! Issues   🔗 Pull requests   ▶ Actions   📁 Projects   📖 Wiki   ⚠ Security   📈 Insights   ⚙ Settings

🔗 master   🔗 1 branch   🏷 0 tags

Go to file   Add file   Code

rodrigofujioka AULA02 - JavaScript, PHP , JSON , Verbos HTTP

js	AULA02 - JavaScript, PHP , JSON , Verbos HTTP
json	AULA02 - JavaScript, PHP , JSON , Verbos HTTP
.gitignore	Initial commit
LICENSE	Initial commit
README.md	Initial commit

Clone with HTTPS   Use SSH

Use Git or checkout with SVN using the web URL.

`https://github.com/rodrigofujioka/clienteservidor`

Open with GitHub Desktop

Download ZIP

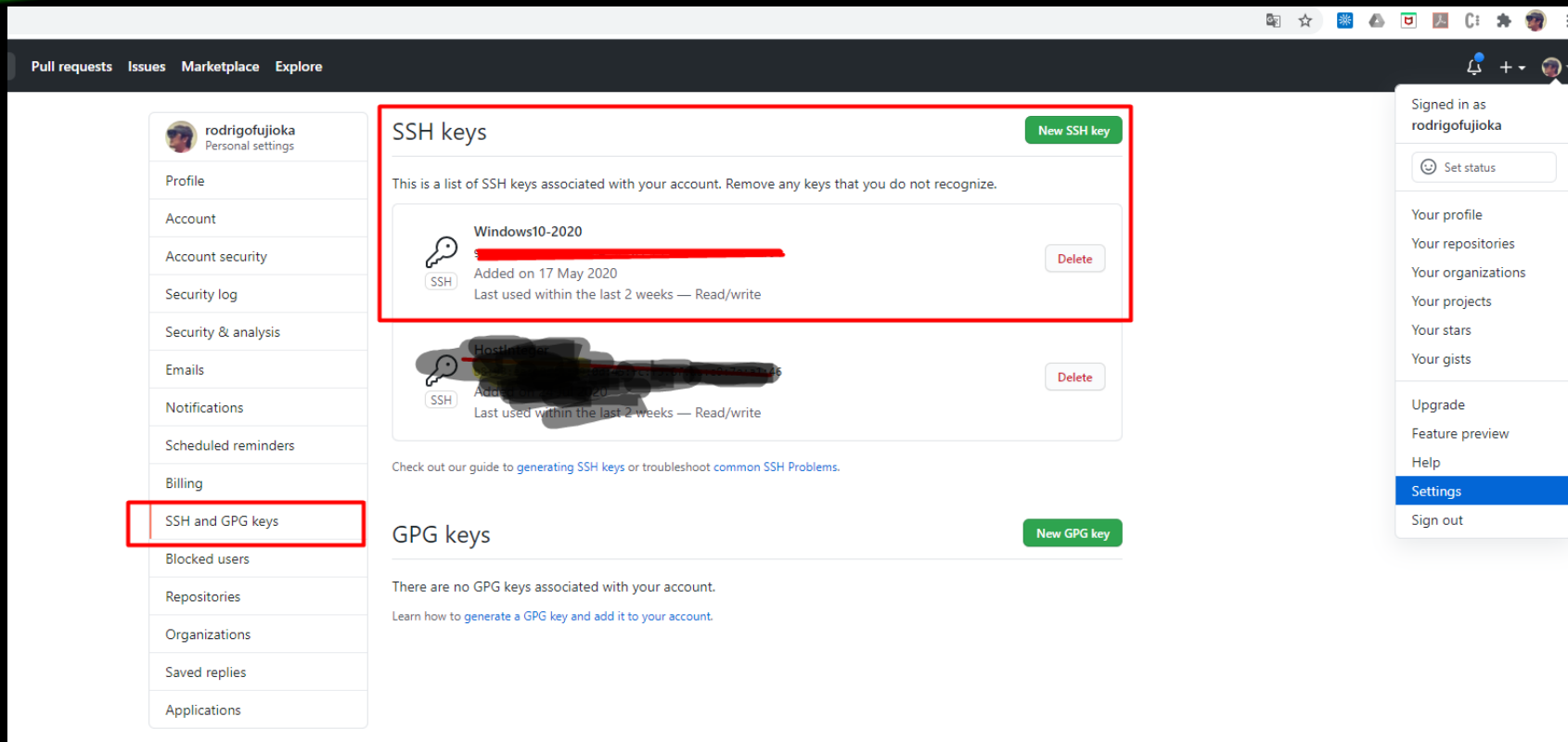
3 hours ago

README.md

clienteservidor



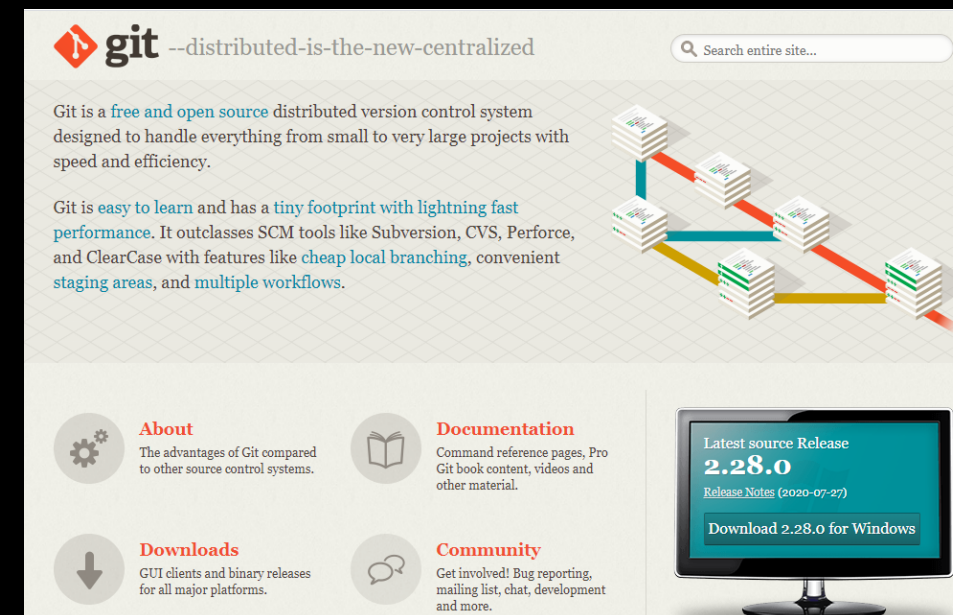
# PARA UTILIZAR SSH



- <https://docs.github.com/pt/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account>

# CLONANDO REPOSITÓRIO

- \$ git clone <https://github.com/rodrigofujioka/clienteservidor.git>  
[precisa da ssh-key configurado no repositório]
- \$ git clone <https://github.com/rodrigofujioka/clienteservidor.git>
- <https://git-scm.com/> <- Você precisa ter o git instalado na sua máquina.



# INICIANDO O PROJETO

- 1 - Quando o projeto for clonado vai ser criada uma pasta com o nome frameworksweb
- 2 - Na pasta que foi criada, pelo terminal ou visual Studio Code vai executar o comando **\$ npm init**
- 3 - Preencha os dados ou clique no enter para os dados padrões serem setados.**

```
package name: (aula01)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /mnt/f/GoogleDrive/academia/docencia/Disciplina/aula01/package.json:
{
  "name": "aula01",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

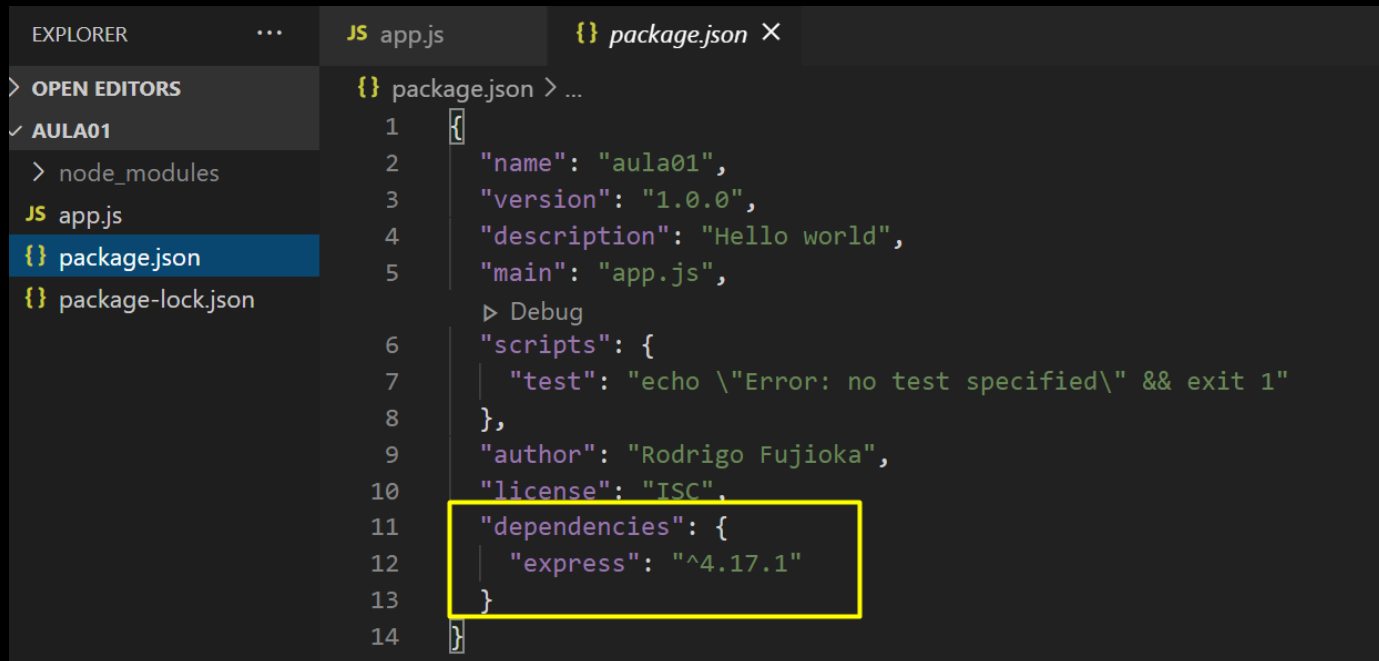
```
sjs\aula01> npm init
```

# INSTALANDO O EXPRESS

- No terminal, na pasta do projeto.

\$ npm install express --save

```
aula01> npm install express --save
```



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows the project structure with 'package.json' selected. The main editor displays the 'package.json' file. A yellow box highlights the 'dependencies' section, which now includes 'express': '^4.17.1'. The file content is as follows:

```
{
  "name": "aula01",
  "version": "1.0.0",
  "description": "Hello world",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Rodrigo Fujioka",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

<https://expressjs.com/pt-br/>

# GET

```
1 // GET method route
2 app.get('/', function(req, res) {
3     res.send('Get Fujioka');
4 });
5
```

- Req (Request)
- Res (Response)

a chamada a `http://localhost:3000`  
vai retornar no navegador **Get Fujioka**

# POST

```
// POST method route  
app.post('/', function (req, res) {  
  res.send('POST Fujioka');  
});
```

- Req (Request)
- Res (Response)



# EXERCÍCIO

- 1 - Crie endpoints que aceitem requisições com PUT , DELETE
- 2 – Crie um endpoint que receba requisições de todos os verbos.

# ALL REQUESTS

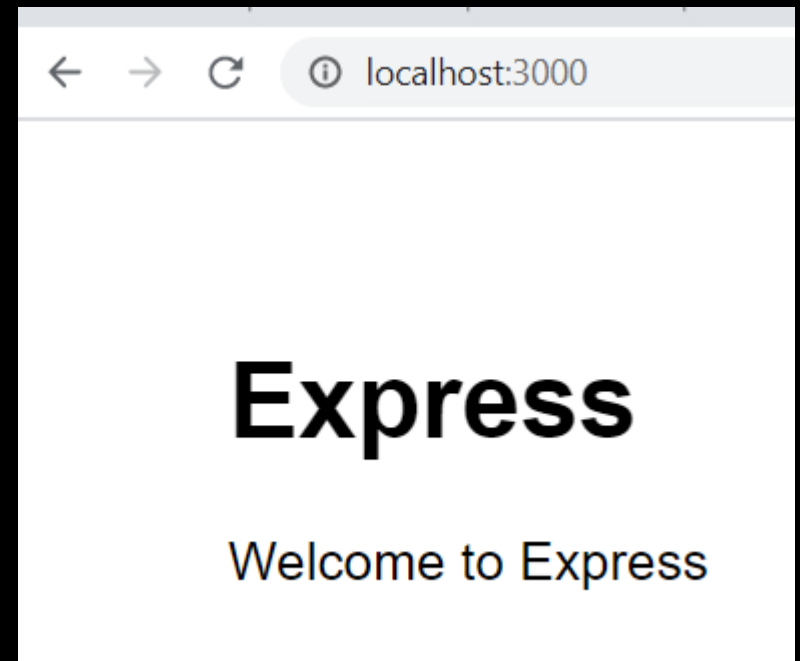
```
app.all('/all', function (req, res, next) {  
  console.log('Recebe requisição de todos ...');  
});
```

# EXPRESS GENERATOR

- \$ npm install express-generator -g
- \$ express -h <- Comando que lista as opções disponíveis.

- 1 - \$ express -view=no-view atendimentoapi
- 2 - \$ cd atendimentoapi
- 3 - \$ npm install
- 4 - \$ npm start
- 5 - Digite no navegador web (<http://localhost:3000>)

Branch: <https://github.com/rodrigofujioka/frameworksweb/releases/tag/1.1.0>



# ORGANIZAÇÃO

## ESTRUTURA CRIADA

- O express generator cria uma estrutura básica para nosso projeto. É uma opção de utilização, sendo possível escolher templates para view.

```

✓ ATENDIMENTOAPI
  > bin
  > node_modules
  ✓ public
    > images
    > javascripts
    > stylesheets
    <> index.html
  ✓ routes
    JS index.js
    JS users.js
  JS app.js
  {} package.json
  {} package-lock.json U

1  var express = require('express');
2  var path = require('path');
3  var cookieParser = require('cookie-parser');
4  var logger = require('morgan');
5
6  var indexRouter = require('./routes/index');
7  var usersRouter = require('./routes/users');
8
9  var app = express();
10
11 app.use(logger('dev'));
12 app.use(express.json());
13 app.use(express.urlencoded({ extended: false }));
14 app.use(cookieParser());
15 app.use(express.static(path.join(__dirname, 'public')));
16
17 app.use('/', indexRouter);
18 app.use('/users', usersRouter);
19
20 module.exports = app;
21
```

# ROTAS

- **Index.js** e **users.js**

# ROUTES

## ✓ ATENDIMENTOAPI

- > bin
- > node\_modules
- ✓ public
  - > images
  - > javascripts
  - > stylesheets
  - <> index.html
- ✓ routes
  - JS index.js
  - JS users.js
- JS app.js
- { } package.json
- { } package-lock.json U

```
1  var express = require('express');
2  var path = require('path');
3  var cookieParser = require('cookie-parser');
4  var logger = require('morgan');
5
6  var indexRouter = require('./routes/index');
7  var usersRouter = require('./routes/users');
8
9  var app = express();
10
11  app.use(logger('dev'));
12  app.use(express.json());
13  app.use(express.urlencoded({ extended: false }));
14  app.use(cookieParser());
15  app.use(express.static(path.join(__dirname, 'public')));
16
17  app.use('/', indexRouter);
18  app.use('/users', usersRouter);
19
20  module.exports = app;
21
```

# EXERCÍCIO

- 1 - Implemente as rotas para criação dos atendimentos na coordenação.
- 2 – Implemente as rotas de forma que seja possível (Cadastrar, Atualizar, Listar e remover atendimentos)

Branch: <https://github.com/rodrigofujioka/frameworksweb/releases/tag/1.2.0>



# MÃO NA MASSA



<https://github.com/rodrigofujioka/frameworksweb>

# PERGUNTAS ?



# ESPECIALIZAÇÕES ?

Especialização em:

**DESENVOLVIMENTO DE  
APLICAÇÕES PARA WEB**

MBA em:

**ARQUITETURA E SOLUÇÕES  
DE TI**



# REFERÊNCIAS

- GITHUB. <https://github.com/rodrigofujioka/frameworksweb>
- GAMA, A. O que é JSON. <https://www.devmedia.com.br/o-que-e-json/23166>. Devmedia, 2011.
- MONZILA(2). JSON . [https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global\\_Objects/JSON](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/JSON). Monzila, 2020.