Programação para Dispositivos Móveis

Profª Letícia Pieper Jandt

80 Horas

UNIFASIPE
CENTRO UNIVERSITÁRIO

Notação JSON

Notação JSON

JSON (JavaScript Object Notation - Notação de Objetos JavaScript) é uma formatação leve de troca de dados. Para seres humanos, é fácil de ler e escrever. Para máquinas, é fácil de interpretar e gerar.

UNIFASIPE
CENTRO UNIVERSITÁRIO

Notação JSON

JSON é em formato texto e completamente independente de linguagem, pois usa convenções que são familiares às linguagens C e familiares, incluindo C++, C#, Java, JavaScript, Perl, Python e muitas outras. Estas propriedades fazem com que JSON seja um formato ideal de troca de dados.

Notação JSON

JSON está constituído em duas estruturas:

Uma coleção de pares nome/valor.

Uma lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma array, vetor, lista ou sequência.

Imagine um *Array* de Textos

```
Array of Strings [5] =
[“Direito”, “ADS”, “Nutrição”,
“Odontologia”, “Jornalismo”]
```

Porem de uma estrutura de Pares organizados em Chave e Valor

Key : Value

Onde Key é o “nome” do campo, e Value o Valor, que pode ter alguns tipos.

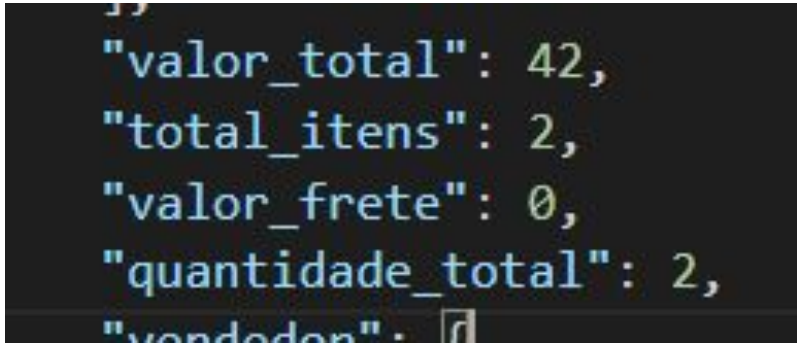
Exemplo:

```
{
   “materia”:“Programação para Dispositivos Móveis”,
   “professor”:“Willian Hübner”,
   “dia”:“Terça-Feira”
}
```

UNIFASIPE
CENTRO UNIVERSITÁRIO

Tipos de Dados

Para as **Chaves**, sempre utilizamos uma *String*, O campo chave precisa ser **ÚNICO** dentro de um objeto.
Quando há mais de um valor para a mesma chave, um novo *array* deve ser formado.



UNIFASIPE
CENTRO UNIVERSITÁRIO

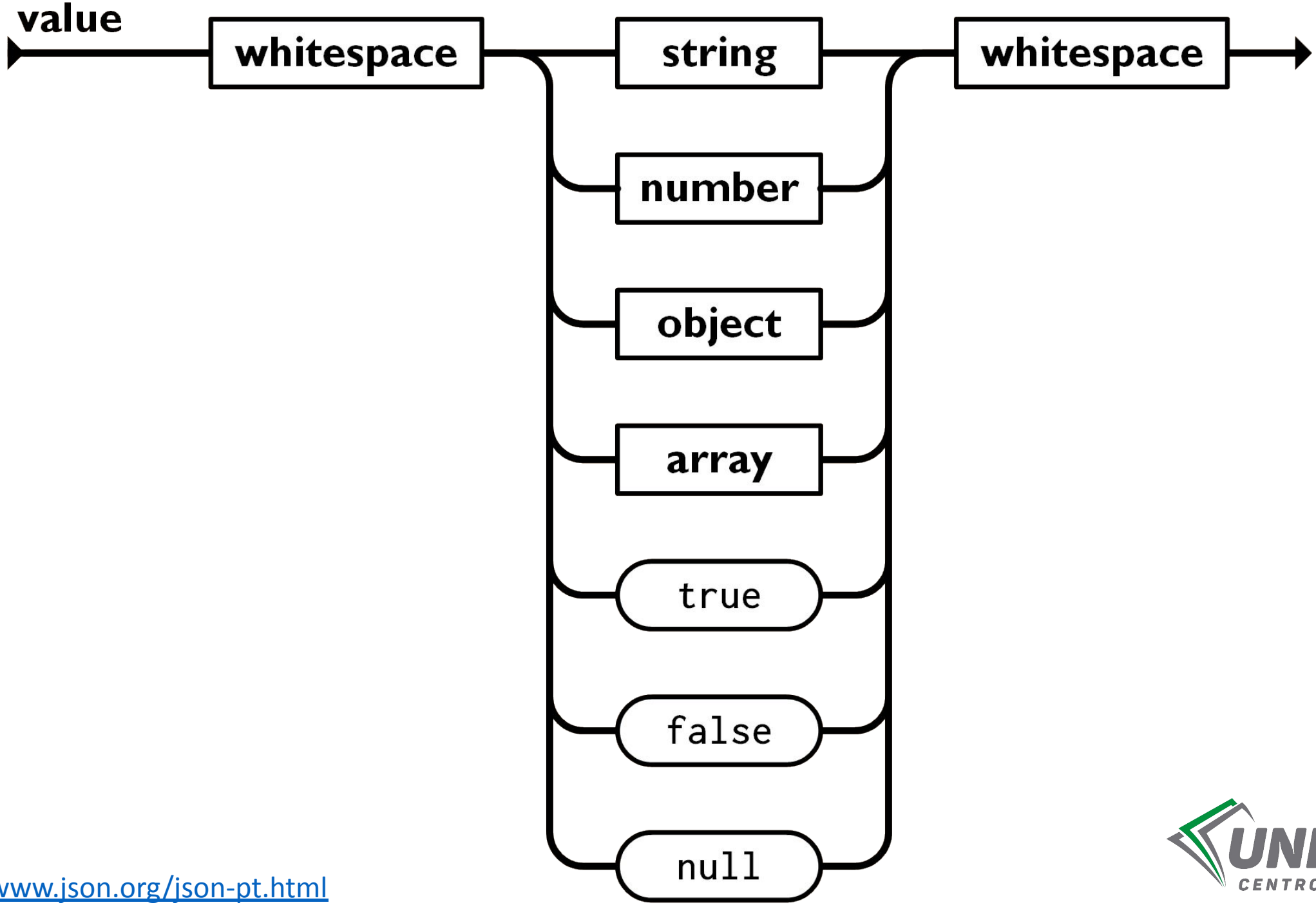
Tipos de Dados

Para os **Valores** podemos resumir objetos JSON em:

- String
- Number
- Boolean
- JsonObject
- JsonArray

```
{
    "nome":"Willian Hübner",
    "semestre":3,
    "regular":true
}
```

UNIFASIPE
CENTRO UNIVERSITÁRIO



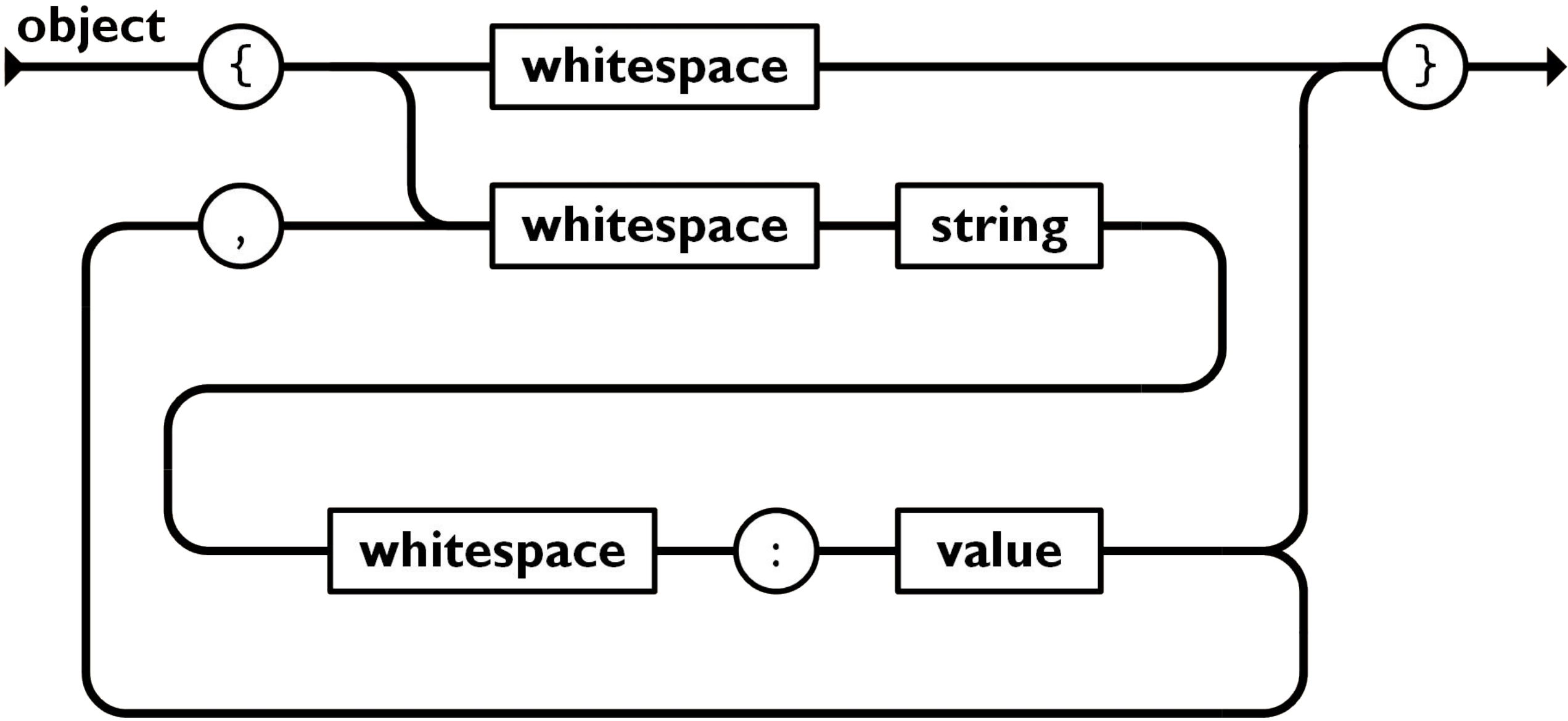
https://www.json.org/json-pt.html

UNIFASIPE
CENTRO UNIVERSITÁRIO

JSONObject

A Estrutura de um Objeto JSON equivale a um Array de pares, iniciado com { e finalizado com }.
Um Objeto JSON pode ser um valor de outro objeto JSON.


```
  {
    "postId": 1,
    "id": 1,
    "name": "id labore ex et quam laborum",
    "email": "Eliseo@gardner.biz",
    "body": "laudantium enim quasi est quidem magnam voluptate ipsam eos\ntempora quo nece
ssitatibus\ndolor quam autem quasi\nreiciendis et nam sapiente accusantium"
  }
```



https://www.json.org/json-pt.html

UNIFASIPE
CENTRO UNIVERSITÁRIO

Objeto de Objeto

```
{
    "nome":"Willian Hübner",
    "semestre":3,
    "regular":true,
    "endereco":{
        "rua":"Avenida da FASIPE",
        "numero":1234,
        "bairro":"ADS 03",
        "cidade":"Sinop",
        "uf":"MT"
    }
}
```

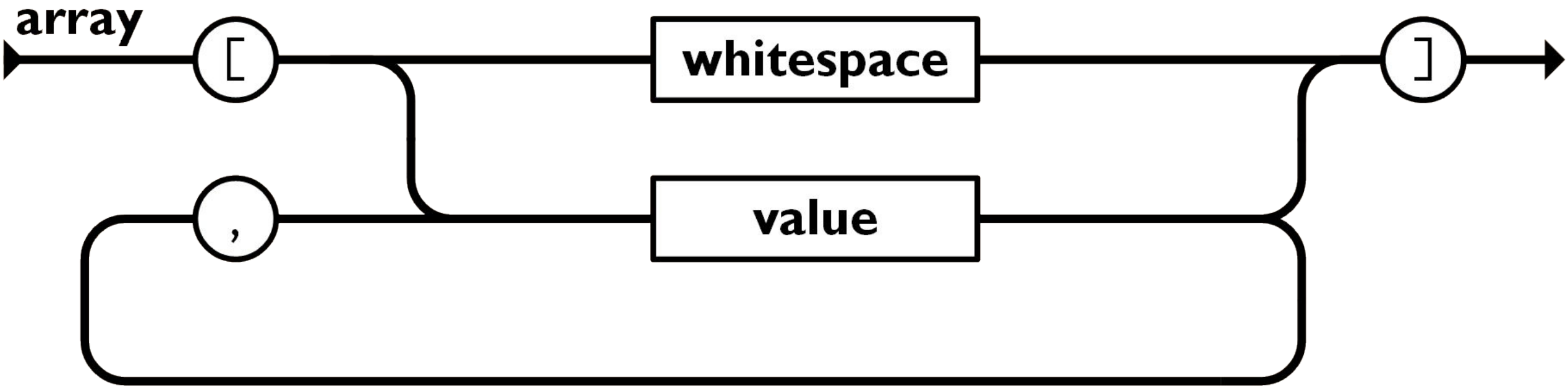
https://www.json.org/json-pt.html

UNIFASIPE
CENTRO UNIVERSITÁRIO

JSON Array

Um JSON Array pode ser a ordenação de vários objetos JSON, separados por , (virgula) e iniciados por [e terminado com].

JSON Array




```
[
  {
    "postId": 1,
    "id": 1,
    "name": "id labore ex et quam laborum",
    "email": "Eliseo@gardner.biz",
    "body": "laudantium enim quasi est quidem magnam voluptate ipsam eos\ntempora quo necessitatib
us\ndolor quam autem quasi\nreiciendis et nam sapiente accusantium"
  },
  {
    "postId": 1,
    "id": 2,
    "name": "quo vero reiciendis velit similique earum",
    "email": "Jayne_Kuhic@sydney.com",
    "body": "est natus enim nihil est dolore omnis voluptatem numquam\net omnis occaecati quod ull
am at\nvoluptatem error expedita pariatur\nnihil sint nostrum voluptatem reiciendis et"
  },
  {
    "postId": 1,
    "id": 3,
    "name": "odio adipisci rerum aut animi",
    "email": "Nikita@garfield.biz",
    "body": "quia molestiae reprehenderit quasi aspernatur\naut expedita occaecati aliquam eveniet
 laudantium\nomnis quibusdam delectus saepe quia accusamus maiores nam est\ncum et ducimus et vero
 voluptates excepturi deleniti ratione"
  }
]
```


UNIFASIPE
CENTRO UNIVERSITÁRIO

Um JSON Array também é um valor JSON.

```json
{
    "nome":"Willian Hübner",
    "semestre":3,
    "regular":true,
    "endereco":{
        "rua":"Avenida da FASIPE",
        "numero":1234,
        "bairro":"ADS 03",
        "cidade":"Sinop",
        "uf":"MT"
    },
    "carros":[
        {
            "marca":"mercedes",
            "modelo":"C180",
            "km":1000
        },
        {
            "marca":"VW motors",
            "modelo":"Fusca",
            "km":8000
        },
        {
            "marca":"Chevrolet",
            "modelo":"Camaro",
            "km":15000
        }
    ]
}
```

UNIFASIPE
CENTRO UNIVERSITÁRIO

Pergunta Rápida, para que serve cada um destes elementos na Notação JSON?

a) [

b) “”

c) {

d) ,

UNIFASIPE
CENTRO UNIVERSITÁRIO

Algumas empresas que realizam a integração com JSON

Create an Activity (createActivity)

Creates a manual activity for an athlete, requires activity:write scope.

POST `/activities`

Parameters

name required String, in form	The name of the activity.
type required String, in form	Type of activity. For example - Run, Ride etc.
start_date_local required Date, in form	ISO 8601 formatted date time.
elapsed_time required Integer, in form	In seconds.
description String, in form	Description of the activity.
distance Float, in form	In meters.
trainer Integer, in form	Set to 1 to mark as a trainer activity.
commute Integer, in form	Set to 1 to mark as commute.

Responses

HTTP code 201	The activity's detailed representation. An instance of DetailedActivity.
HTTP code 4xx, 5xx	A Fault describing the reason for the error.

HTTPie | Java | Obj-C | JavaScript | C# | Python

```
$ http POST "https://www.strava.com/api/v3/activities" name='value' type='value' start_
```

Sample Response


```
{
  "id" : 123456778928065,
  "resource_state" : 3,
  "external_id" : null,
  "upload_id" : null,
  "athlete" : {
    "id" : 12343545645788,
    "resource_state" : 1
  },
  "name" : "Chill Day",
  "distance" : 0,
  "moving_time" : 18373,
  "elapsed_time" : 18373,
  "total_elevation_gain" : 0,
  "type" : "Ride",
  "start_date" : "2018-02-20T18:02:13Z",
  "start_date_local" : "2018-02-20T10:02:13Z",
  "timezone" : "(GMT-08:00) America/Los_Angeles",
  "utc_offset" : -28800,
  "achievement_count" : 0,
  "kudos_count" : 0,
  "comment_count" : 0,
  "athlete_count" : 1,
  "photo_count" : 0,
  "map" : {
    "id" : "a12345678908766",
    "polyline" : null,
    "resource_state" : 3
  },
  "trainer" : false,
  "commute" : false,
  "manual" : true,
  "private" : false,
  "flagged" : false,
  "gear_id" : "b453542543",
  "from_accepted_tag" : null,
  "average_speed" : 0,
  "max_speed" : 0,
  "device_watts" : false,
  "has_heartrate" : false,
  "pr_count" : 0,
  "total_photo_count" : 0,
  "has_kudoed" : false,
  "workout_type" : null,
  "description" : null,
```

Before you start developing with the Directions API, review the authentication requirements (you need an API key) and the API usage and billing information (you need to enable billing on your project).

Directions requests

A Directions API request takes the following form:

```
https://maps.googleapis.com/maps/api/directions/outputFormat?parameters
```

where `outputFormat` may be either of the following values:

- `json` (recommended) indicates output in JavaScript Object Notation (JSON)
- `xml` indicates output as XML

Note: URLs must be properly encoded to be valid and are limited to 8192 characters for all web services. Be aware of this limit when constructing your URLs.

HTTPS or HTTP

Security is important and HTTPS is recommended whenever possible, especially for applications that include sensitive user data, such as a user's location, in requests. Using HTTPS encryption makes your application more secure, and more resistant to snooping or tampering.

If HTTPS is not possible, to access the Directions API over HTTP, use:

```
http://maps.googleapis.com/maps/api/directions/outputFormat?parameters
```

Default response

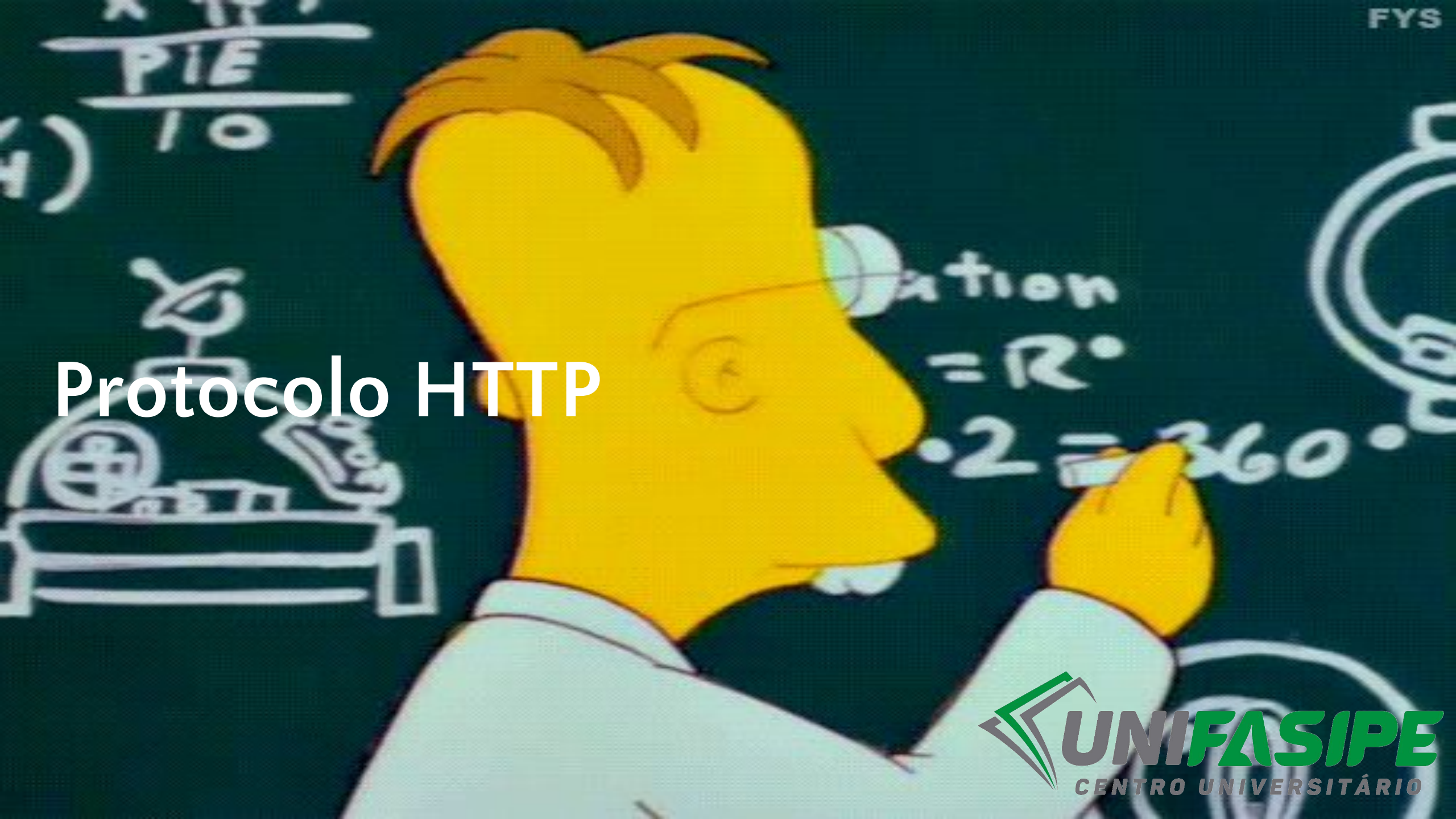
```
Status: 200 OK
```


```
{
  "total_count": 2,
  "artifacts": [
    {
      "id": 11,
      "node_id": "MDg6QXJ0aWZhY3QxMQ==",
      "name": "Rails",
      "size_in_bytes": 556,
      "url": "https://api.github.com/repos/octo-org/octo-docs/actions/artifacts/11",
      "archive_download_url": "https://api.github.com/repos/octo-org/octo-docs/actions/artif
      "expired": false,
      "created_at": "2020-01-10T14:59:22Z",
      "expires_at": "2020-01-21T14:59:22Z"
    },
    {
      "id": 13,
      "node_id": "MDg6QXJ0aWZhY3QxMw==",
      "name": "",
      "size_in_bytes": 453,
      "url": "https://api.github.com/repos/octo-org/octo-docs/actions/artifacts/13",
      "archive_download_url": "https://api.github.com/repos/octo-org/octo-docs/actions/artif
      "expired": false,
      "created_at": "2020-01-10T14:59:22Z",
      "expires_at": "2020-01-21T14:59:22Z"
```


Notes

- Works with GitHub Apps

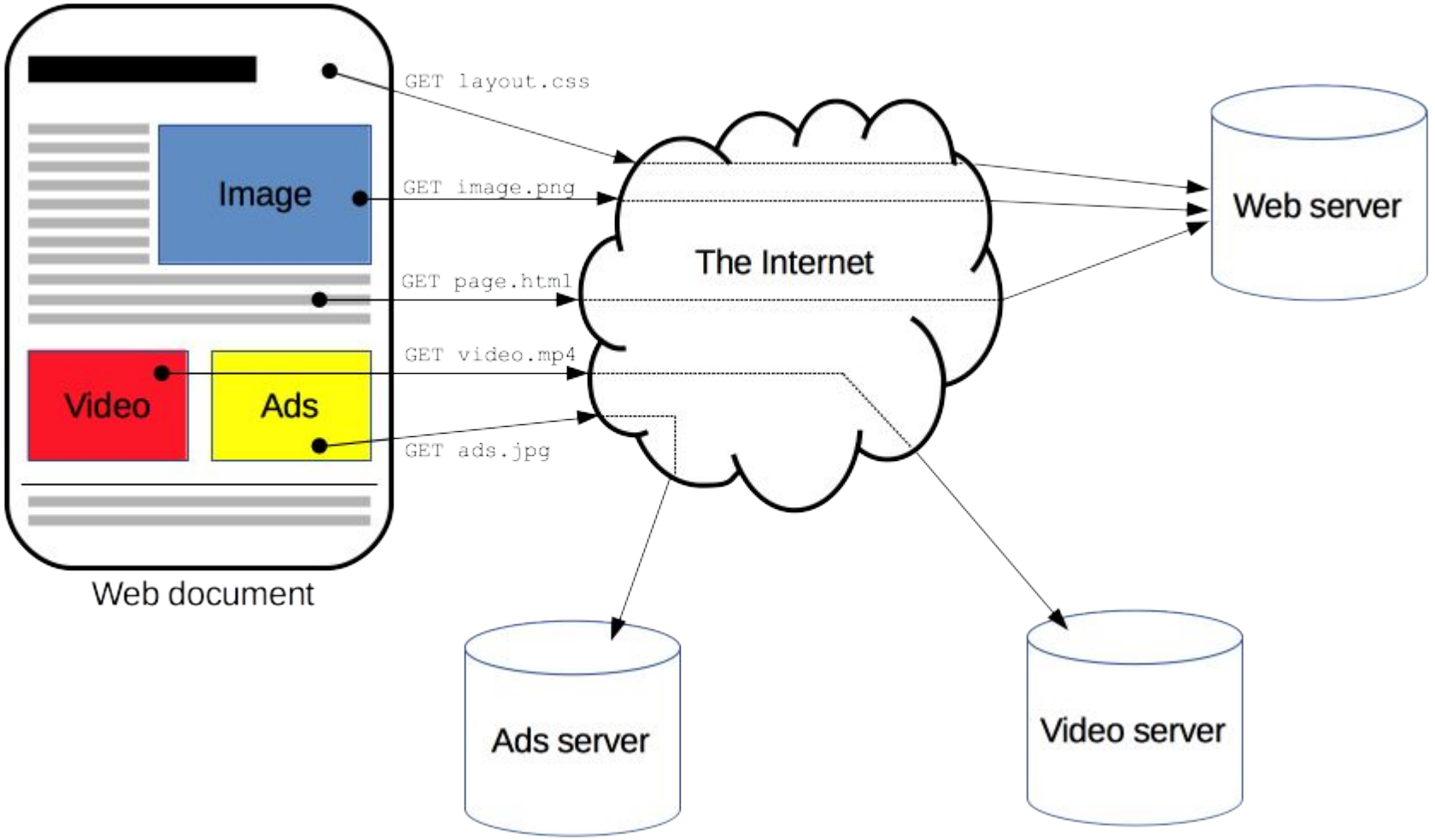
Protocolo HTTP



UNIFASIPE
CENTRO UNIVERSITÁRIO

Protocolo HTTP

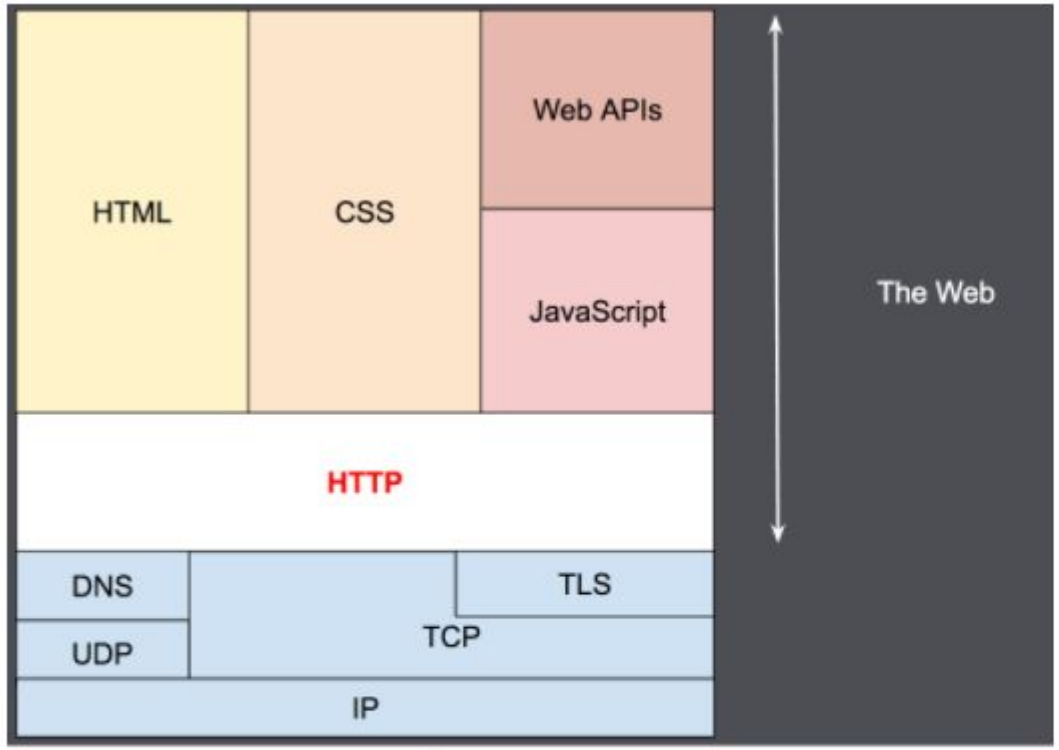
HTTP é um protocolo que permite a obtenção de recursos, tais como documentos HTML. É a base de qualquer troca de dados na Web e um protocolo cliente-servidor, o que significa que as requisições são iniciadas pelo destinatário, geralmente um navegador da Web.


GET layout.css
Image
GET image.png
GET page.html
Video
Ads
GET video.mp4
GET ads.jpg
Web document
The Internet
Web server
Ads server
Video server

Protocolo HTTP

Clientes e servidores se comunicam trocando mensagens individuais (em oposição a um fluxo de dados). As mensagens enviadas pelo cliente, geralmente um navegador da Web, são chamadas de solicitações (requests), ou também requisições, e as mensagens enviadas pelo servidor como resposta são chamadas de respostas (responses).

Protocolo HTTP

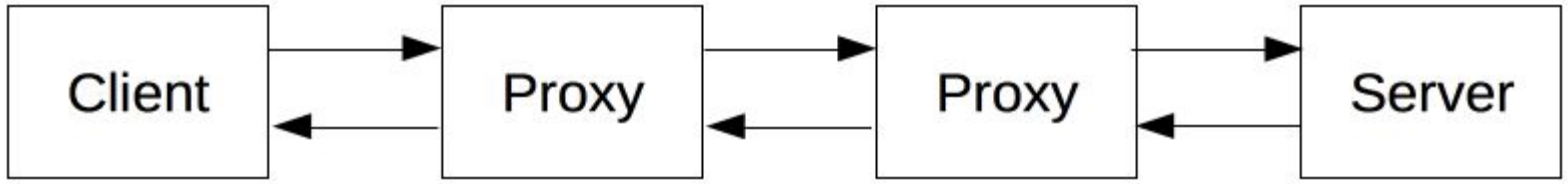


Projetado no início da década de 1990, o HTTP é um protocolo extensível que evoluiu ao longo do tempo. É um protocolo de camada de aplicação que é enviado sobre TCP, ou em uma conexão TCP criptografada com TLS, embora qualquer protocolo de transporte confiável possa, teoricamente, ser usado. Devido à sua extensibilidade, ele é usado para não apenas buscar documentos de hipertexto, mas também imagens e vídeos ou publicar conteúdo em servidores, como nos resultados de formulário HTML (veja os elementos `<html>` e `<form>`). O HTTP também pode ser usado para buscar partes de documentos para atualizar páginas da Web sob demanda.

https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview

UNIFASIPE
CENTRO UNIVERSITÁRIO

Request – Response



Entre a solicitação e a resposta existem várias entidades, designadas coletivamente como proxies, que executam operações diferentes e atuam como *gateways* (intermediários) ou caches, por exemplo

https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview

UNIFASIPE CENTRO UNIVERSITÁRIO

Entidades no HTTP

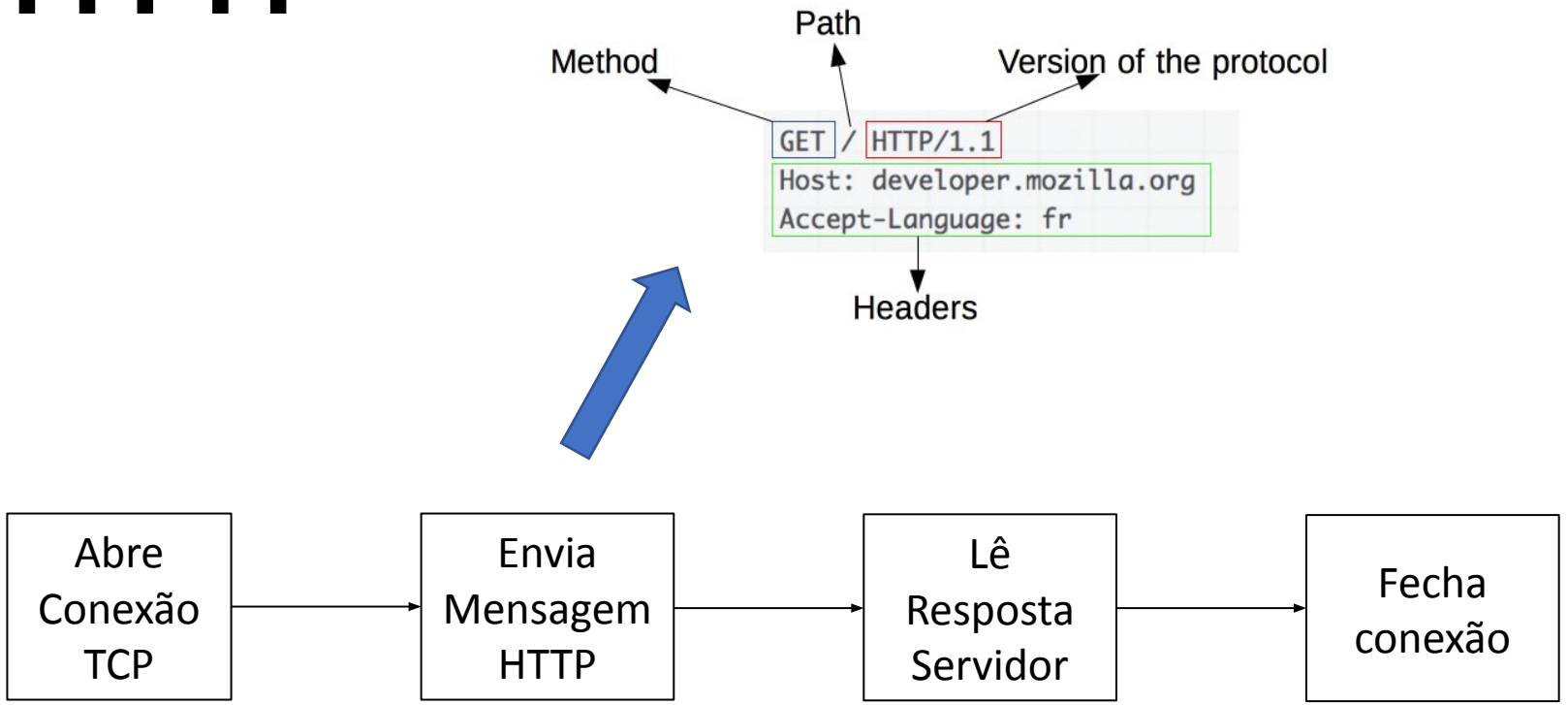
Usuário : user-agent

Servidor

Proxy:

- cacheamento (o *cache* pode ser público ou privado, como o *cache* dos navegadores)
- filtragem (como um *scanner* de antivírus, controle de acesso, etc)
- balanceamento de carga (para permitir que vários servidores possam responder a diferentes requisições)
- autenticação (para controlar quem tem acesso aos recursos)
- autorização (para controlar quem tem acesso a determinada informação)
- registro de informação (permite o armazenamento de informações de histórico)

Fluxo HTTP


Path
Method
Version of the protocol
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
Headers
Abre Conexão TCP
Envia Mensagem HTTP
Lê Resposta Servidor
Fecha conexão

Métodos HTTP

GET
POST
PUT
DELETE

HTTP Responses

200 – OK
400 – Bad Request
404 – Not Found
405 – Method Not Allowed

https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers

Headers

Authorization
Contém as credenciais para autenticar um User-Agent com o servidor.

ETag
É um validador, uma string única identificando a versão do recurso. Requisições condicionais usando If-Match e If-None-Match usam esse valor para modificar o comportamento da requisição.

If-Match
Faz a requisição condicional e aplica o método apenas se o recurso armazenado corresponder a uma das ETags fornecidas.

If-None-Match
Faz a requisição condicional e aplica o método apenas se o recurso armazenado não corresponder a nenhuma das ETags fornecidas. É usado para atualizar caches (para requisições seguras), ou para prevenir o upload de um novo recurso quando este já existe.

Accept
Informa ao servidor sobre os tipos de dados que podem ser enviados de volta. Isto é MIME-type.

Accept-Charset
Informa ao servidor sobre qual conjunto de caracter o cliente é capaz de entender.

Access-Control-Allow-Origin
Indica se a resposta pode ser compartilhada.

Access-Control-Allow-Credentials
Indica se a resposta a requisição pode ou não ser exposta quando a flag de crendenciais é verdadeira.

UNIFASIPE
CENTRO UNIVERSITÁRIO

Exemplos de Requests

```
axios.get('https://api.github.com/users/' +
username) .then(function(response){
console.log(response.data); // ex.: { user:
'Your User'} console.log(response.status);
// ex.: 200 });
```

```
begin
  TRequest.New.BaseURL('http://localhost:8888/users')
    .Accept('application/json')
    .Get;
end;
```

```
$.ajax({
        type:'GET',
        url: "https://apiadvisor.climatempo.com.br/api/v1/locale/city?name="+sessionS
torage.getItem('cidade')+"&state="+sessionStorage.getItem('uf')+"&token=186af1cb66fcd
929347475d3d720b297",
        crossDomain:true,
        success: function(data){
            if (!data.errors){
                    sessionStorage.setItem('cod_cidade_dev', data[0].id);

                    consultaClima();
                }
            },
        error: function (request, status, erro) {
            console.log('erro:'+request);
        }
    });
```

UNIFASIPE
CENTRO UNIVERSITÁRIO