



# Meiuca

Módulo 5 - Aula 6 - Hard



Fonte: Pixabay




Todos os direitos reservados  
©2022 Resilia Educação

# Problema da aula

---

Temos requisições passando dados, mas não conseguimos enxergar/utilizar esses dados passados ainda. Será que existe uma maneira de isso acontecer?

A black rotary telephone handset is positioned on the left side of the frame, with its cord extending across the bottom and right. The background is a solid dark gray. The text "Comunicação com API" is centered in a bold, yellow, sans-serif font.

# Comunicação com API



# Requisições



Todos os direitos reservados  
©2022 Resilia Educação

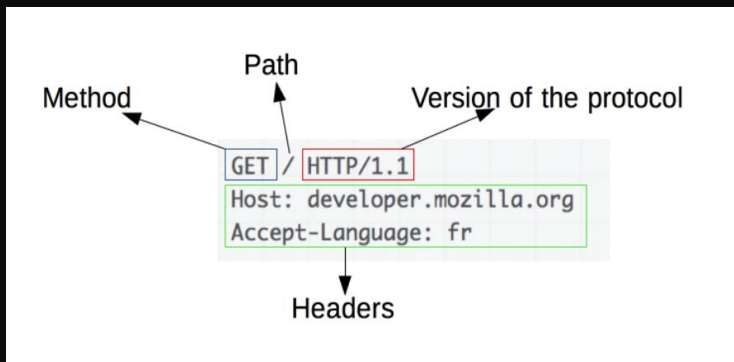
# Requisição

---

Requisições são a base da comunicação na web. Quando falamos de **API's**, os **clientes** fazem **requisições** e a **API** é responsável por **respondê-la** de acordo com o recurso solicitado.

# Anatomia de requisição HTTP (GET)

Abaixo, temos uma requisição que utiliza o método **GET** com seus campos:



# Anatomia de requisição HTTP (POST)

Abaixo, temos uma requisição que utiliza o método **POST**:

Quais os **campos presentes**?

```
POST / HTTP/1.1
Host: localhost:8080
User-Agent: insomnia/2020.5.2
Content-Type: application/json
Accept: */*
Content-Length: 42

{
  "nome": "Resilia",
  "curso": "Webdev"
}
```

# Objeto requisição (express)

No express uma **rota** recebe uma **callback de ativação**, que recebe como parâmetros os objetos requisição e resposta (**req, resp**).

```
app.get('/tarefas', (req, resp) => {  
    resp.send('Rota ativada: Tarefas sendo consultadas');  
});
```



# Objeto requisição (express)

O objeto requisição (**req**) contém diversos **métodos e atributos**. No caso de uma requisição com método **POST**, queremos ter acesso ao **body** (corpo), para **acessar os dados**.

No entanto, ao acessarmos esse atributo ...



# Middlewares



# Express - Middlewares

---

**Middlewares** são complementos intermediários. Eles são adicionados ao express utilizando o **método use**. Suas funções são principalmente:

- Manipular a requisição antes de atingir as rotas
- Adicionar novos método/funcionalidades ao express
- Tratar erros

# Body-parser

---

O **body-parser** é um **middleware** responsável por **processar o corpo das requisições**, seguindo um padrão definido, como por exemplo, **JSON**.

# Utilizando o body-parser via express

O *body-parser* está disponível através do próprio *express* e para comunicações utilizando *JSON*, podemos utilizar o método *express.json()*. Instanciar um objeto disponível no próprio *express*.

Após isso, *importamos o módulo* e passamos o parser desejado para o *método use do express*. No exemplo, temos a configuração do *body-parser para JSON*.

```
const express = require('express');  
const app = express();  
app.use(express.json());
```

# Atividade: POST & body-parser

---

Utilizando o que desenvolvemos até aqui da TODO app api, faça o que se pede:

- No arquivo do servidor, adicione o body-parser como middleware (com o método use do express), para processar requisições com corpo JSON
- Implemente nas rotas de POST dos controllers uma verificação do corpo (body) do objeto de requisição. Para cada valor recebido faça um console.log simples.

**Obs:** Utilize o insomnia/postman para testar as novas rotas criadas, utilizando o método POST, com um JSON sendo passado no corpo da Requisição





# Atividade em pares

