APIS WEB E ROTEAMENTO

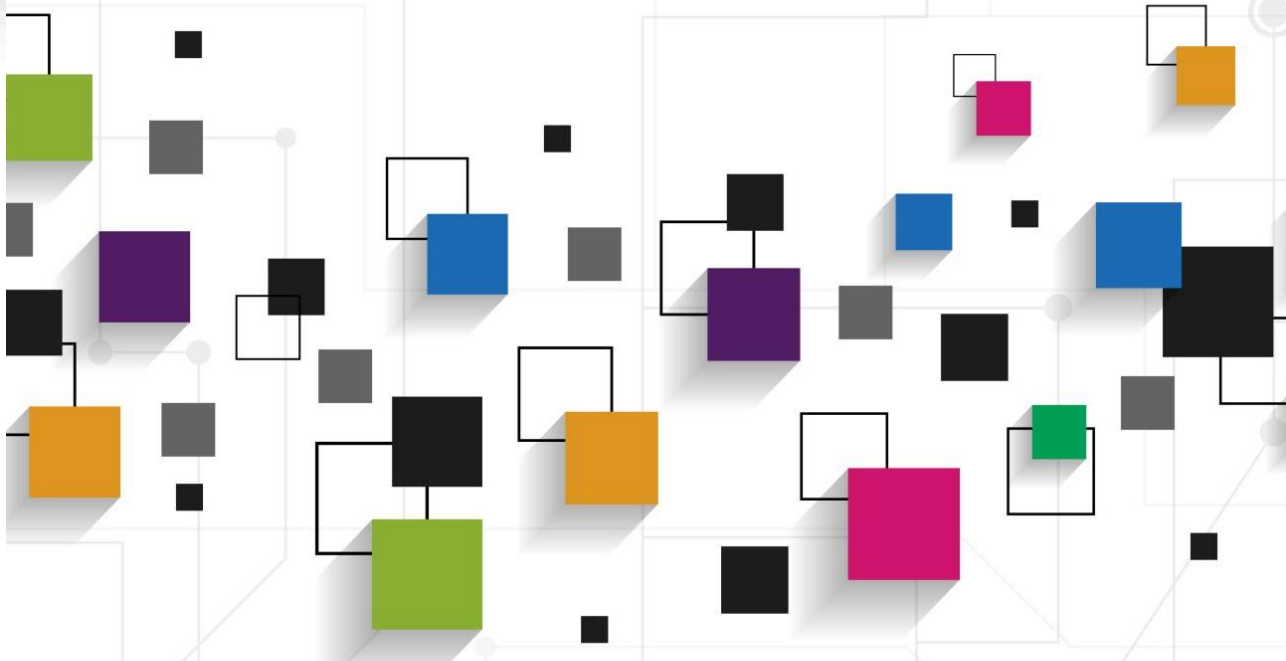
prof. Me. Hélio Esperidião

Arquitetura de Software

- Define a maneira como os componentes de software estão organizados e interagem entre si para atender aos requisitos funcionais e não funcionais do sistema.
- A arquitetura de software é responsável por fornecer uma visão geral do sistema, incluindo seus principais elementos, suas propriedades e as relações entre eles.

MVC

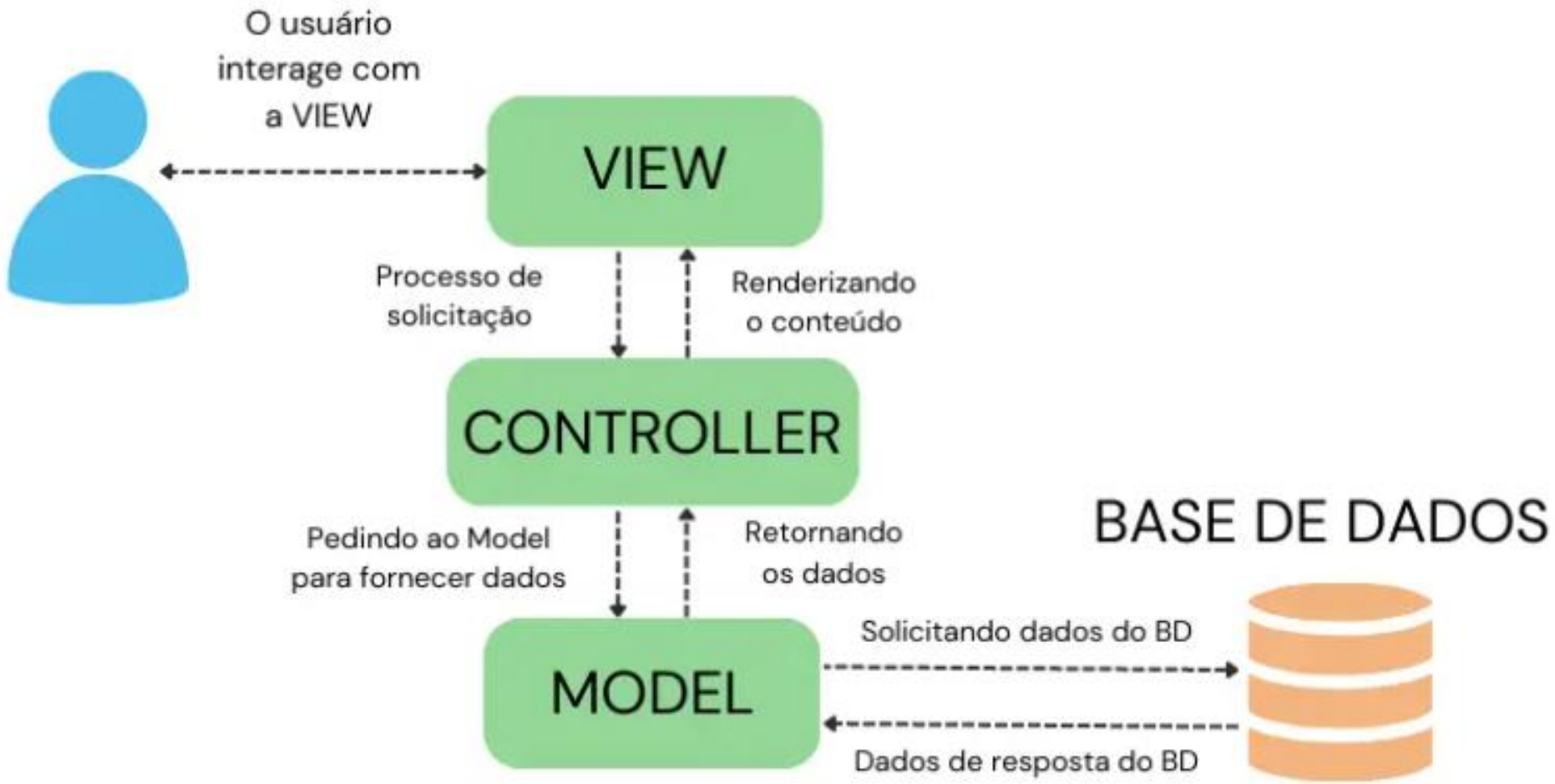
- MVC (Model-View-Controller) é um padrão arquitetural amplamente utilizado no desenvolvimento de software, especialmente em aplicações web e aplicativos para dispositivos móveis.



Padrão MVC

- Modelo
  - Todas as classes da aplicação ficarão armazenadas nesse diretório
- Visualização
  - Todos os arquivos de front-end ficarão armazenados nesse diretório.
- Controle
  - Arquivos que recebem dados do front-end, utilizam as classes e enviam respostas para o front-end.

Arquitetura MVC



Frameworks que podem utilizar MVC

- **Ruby on Rails**: Framework web escrito em Ruby, conhecido por sua forte aderência ao MVC na organização do código.
- **Laravel**: Framework web em PHP que segue o padrão MVC, facilitando a organização e manutenção do código.
- **Django**: Framework web em Python que utiliza o padrão MVC para separar as responsabilidades de forma clara e escalável.
- **Spring**: Framework de aplicação Java que emprega o padrão MVC, proporcionando uma separação eficaz entre a lógica de negócios e a interface do usuário.
- **AngularJS**: Framework JavaScript que segue a variante MVVM (Model-View-ViewModel), uma adaptação do padrão MVC que simplifica a ligação de dados entre a view e o modelo.
- **React**: Biblioteca JavaScript para construção de interfaces de usuário, embora siga uma arquitetura fluxo unidirecional, é comum sua utilização em conjunto com o padrão MVC para organizar e modularizar o código de forma eficiente.

JSON - JavaScript Object Notation

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.

É baseado em um subconjunto da linguagem de programação JavaScript, Standard ECMA-262 3a Edição -Dezembro - 1999.

JSON

01

JSON é em formato texto e completamente independente de linguagem

02

Formato ideal de troca de dados

03

é um formato compacto, de padrão aberto independente, de troca de dados simples e rápida (parsing) entre sistemas

ESTRUTURA

Uma coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um *object*, record, struct, dicionário, hash table, keyed list, ou arrays associativas.

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

Exemplos de json

{"**altura**":57.7,"**largura**":55}

{"**nome**":"helio","**cargo**":"professor"}

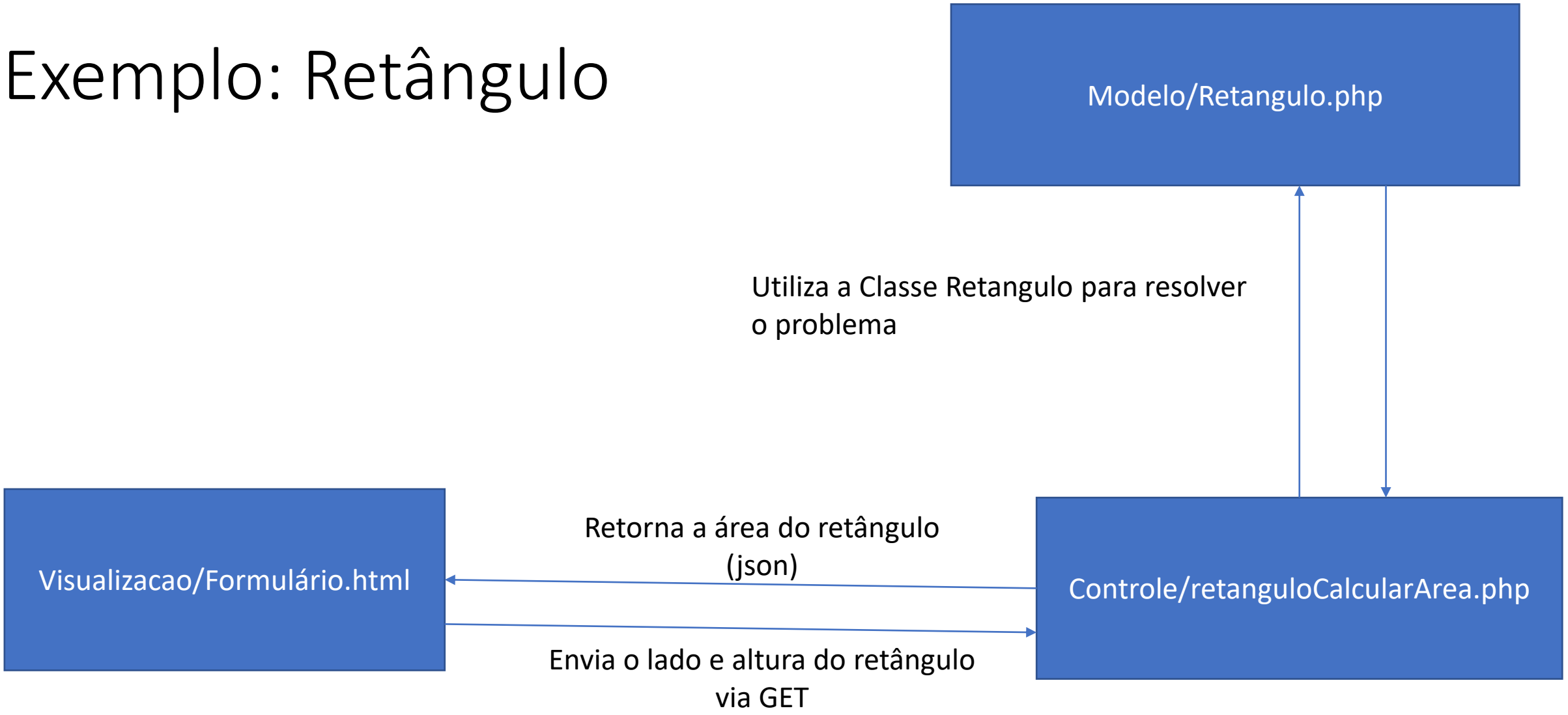
Aplicações modernas.

Tipicamente em uma aplicação moderna o frontend que não utiliza o get envia um json com os dados para um controle que utiliza as classes necessárias para resolver o problema.

O controle também responde para o front-end um json.

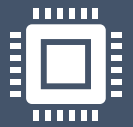
Se o front trabalhar com o método get tipicamente os dados são passados na url.

Exemplo: Retângulo



API - *Application Programming Interface*

API significa Interface de Programação de Aplicações. É um conjunto de definições e protocolos que permite a comunicação entre diferentes softwares.



Uma API define como diferentes componentes de software devem interagir uns com os outros.

Uma API pode ser entendida como uma ponte que permite que diferentes sistemas, aplicativos ou serviços se comuniquem e compartilhem recursos e funcionalidades entre si.

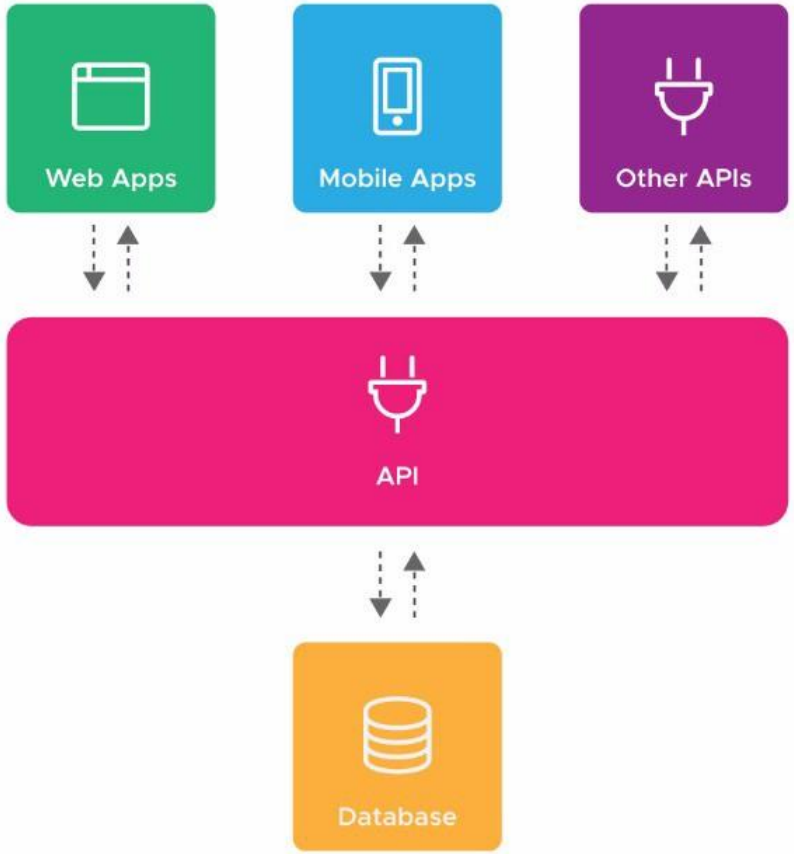
APIs

A sigla API (Application Programming Interface) que, traduzida para o português, pode ser compreendida como uma interface de programação de aplicação.

API é um conjunto de normas que possibilita a comunicação entre plataformas através de uma série de padrões e protocolos.

Por meio de APIs, desenvolvedores podem criar novos softwares e aplicativos capazes de se comunicar com outras plataformas

Esquema de funcionamento de apis



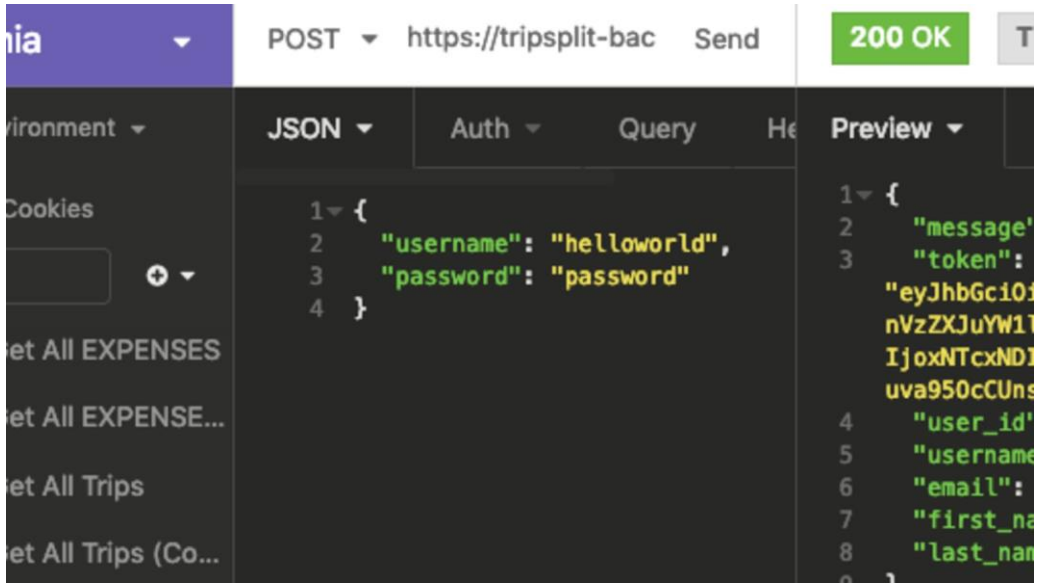
apis interessantes.

- https://api.adviceslip.com/advice

- https://deividfortuna.github.io/fipe/
  - https://parallelum.com.br/fipe/api/v1/carros/marcas
  - https://parallelum.com.br/fipe/api/v1/carros/marcas/59/modelos

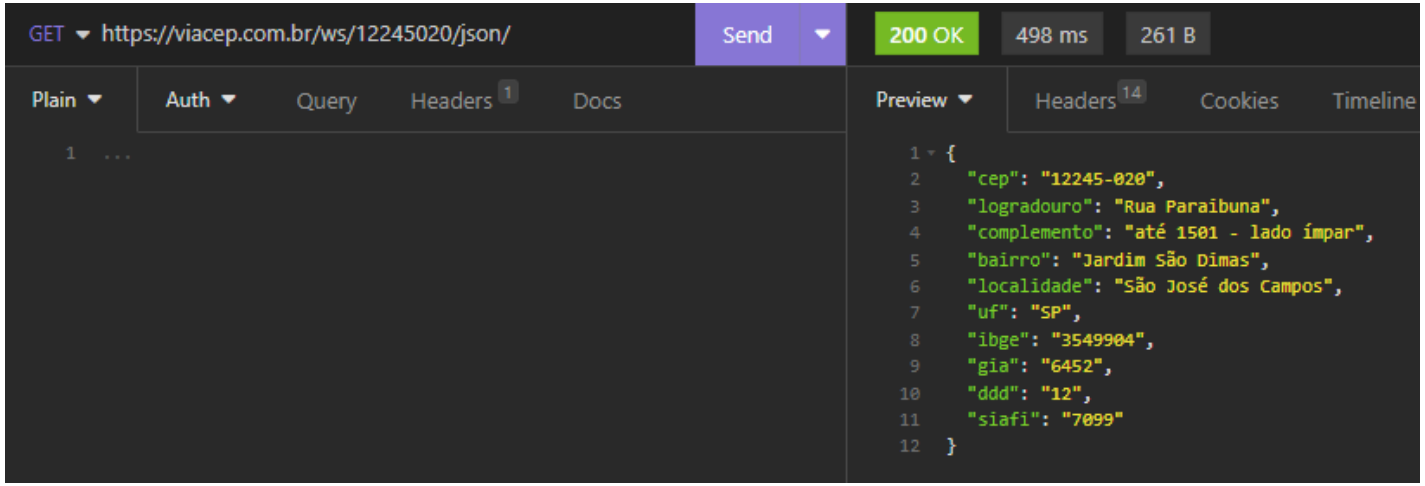
insomnia

- É um software que nos permite fazer requisições à APIs. Ajuda no desenvolvimento do Backend (ex.: testar as chamadas para cada endpoint sem necessitar de um Frontend)
- Download free:
  https://insomnia.rest/



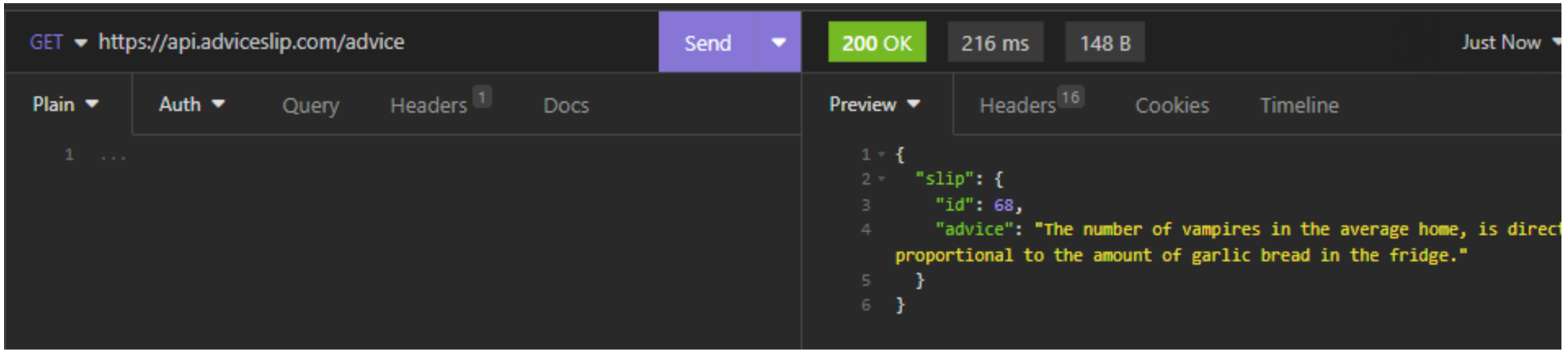
Via cep: https://viacep.com.br/ws/12245020/json/

- Observe que a ferramenta envia um get para o endereço:
  - https://viacep.com.br/ws/12245020/json/
- Observe o formato da url, **não** é utilizado o padrão "? &" para identificar as variáveis e dados que serão enviados para o servidor.
- É possível notas que o cep é enviado na url.



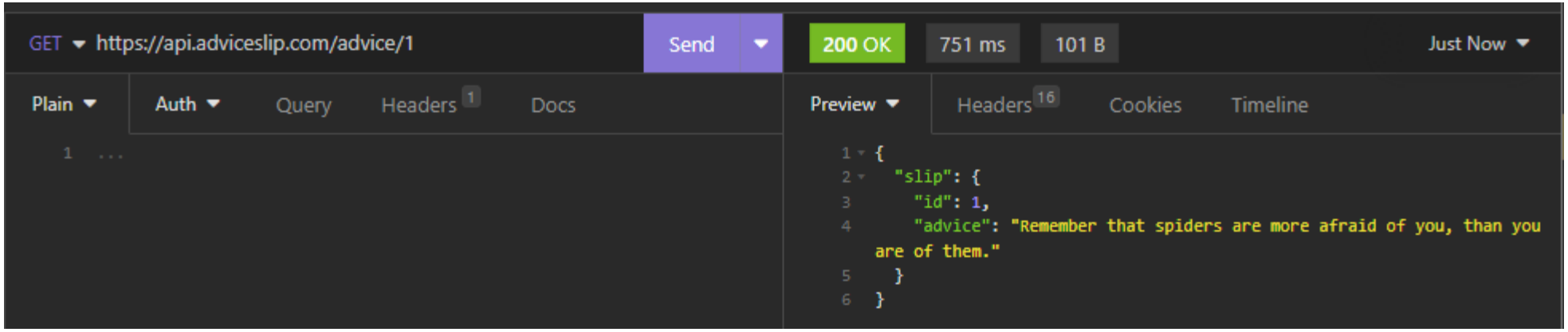
Testando apis:

- Acesse a URL:
  - https://api.adviceslip.com/advice



https://api.adviceslip.com/advice/1

- Observe que é enviado o número 1 pela a url, é possível notar que o 1 é referente a uma chave primária em um banco de dados e que você pode escolher outros números correspondentes a concelhos diferentes.



Testar Minhas APIS

- Exemplo API Retangulo.
  - Classe:
    - Retangulo.php
  - Controles:
    - retangulo_CalcularArea.php
    - retangulo_CalcularPerimetro.php
    - retangulo_CalcularDiagonalPrincipal.php

Classe Retangulo.php.
Considerar que Gets e Sets foram programados

```
class Retangulo{
    private $base;
    private $altura;
    public function calcularArea(){
        return ($this->altura * $this->base);
    }
    public function calcularDiagonal(){
        return sqrt(pow($this->altura, 2) + pow($this->base, 2));
    }
    public function calcularPerimetro(){
        return ($this->altura * 2 + $this->base * 2);
    }
}
```

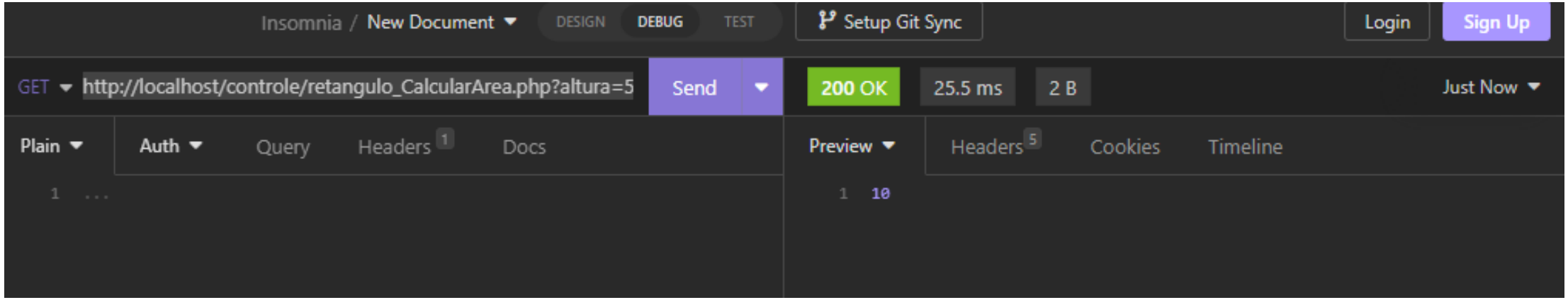
retangulo_CalcularArea.php

```php
<?php
    require_once "../modelo/Retangulo.php";
    $altura = $_GET['altura'];
    $base   = $_GET['base'];

    $r1 = new Retangulo();

    $r1->setAltura($altura);
    $r1->setBase($base);
    $area = $r1->calcularArea();

    header("HTTP/1.1 200 OK");
    echo $area;
?>
```

- header("HTTP/1.1 200 OK");
  - Significa que o processamento e resposta estão ok. “deu tudo certo com a requisição do usuário”

Testar no insominia

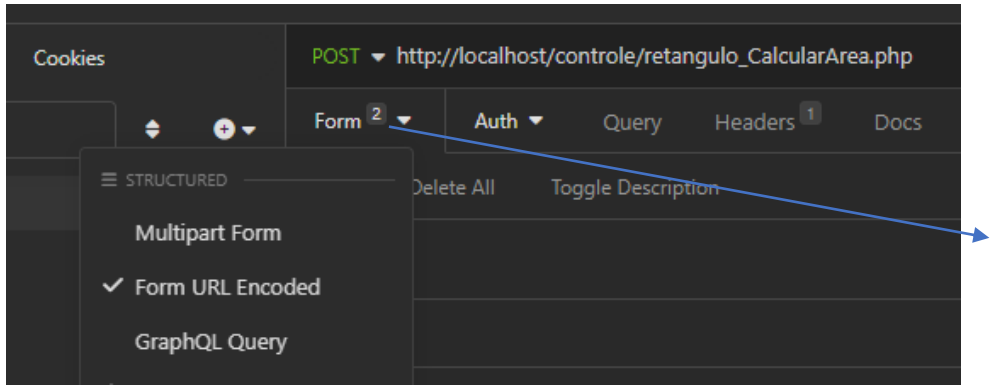
- http://localhost/controle/retangulo_CalcularArea.php?altura=5&base=2



Teste utilizado o método post.
Alterar o arquivo: retangulo_CalcularArea.php

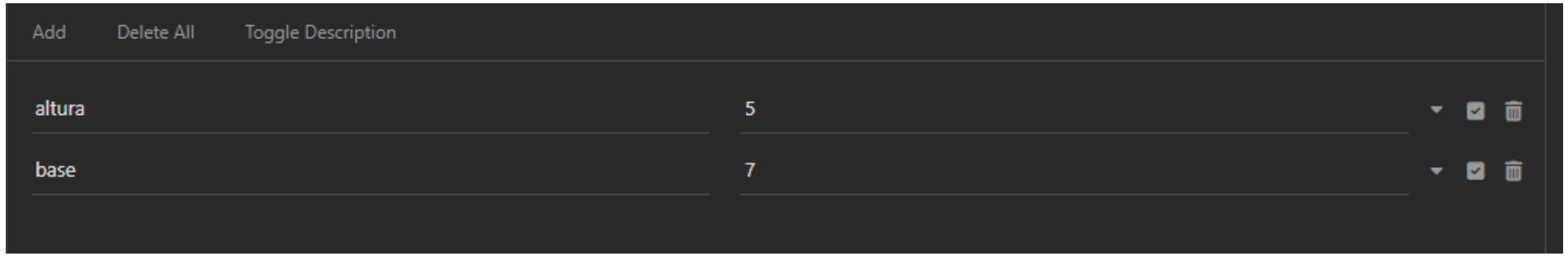
```
<?php
   require_once "../modelo/Retangulo.php";
   $altura = $_POST['altura']; //Trocou para post
   $base   = $_POST['base']; //Trocou para post

   $r1 = new Retangulo();

   $r1->setAltura($altura);
   $r1->setBase($base);
   $area = $r1->calcularArea();

   header("HTTP/1.1 200 OK");
   echo $area;
?>
```

Caso seja enviado um post. Configure a ferramenta

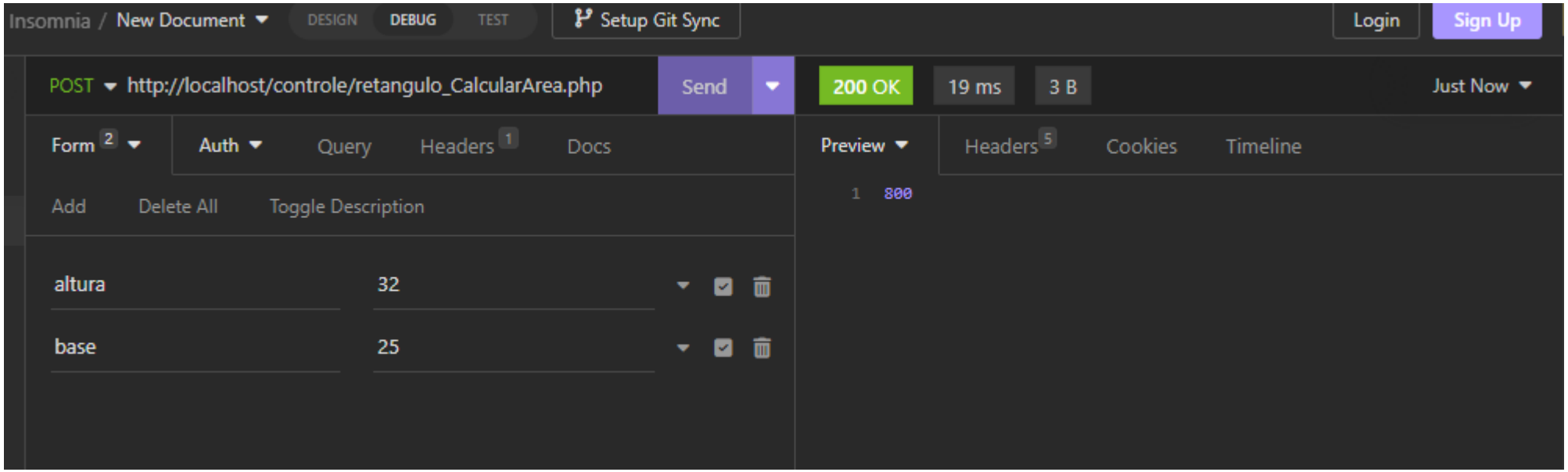


Click nessa posição e escolha a opção “multipart form”

Adicione os “names” que virão do formulário e seus valores



Exemplo teste post:



Apis e Aplicações Modernas

- Tipicamente quando construímos APIs web um dos objetivos é separar completamente o front-end do black end.
- Utilizando O MVC é possível separar o front-end do back end.
- Muitos desenvolvedores tem preferido trabalhar com o JSON para comunicação entre o back-end(POST) e o front-end.
- Os fronts (sites, televisores, carros, dispositivos eletronicos) dados para o back que responde com um JSON.
- Isso garante um isolamento ainda maior do front-end.

Rotas

Rotas na web

Uma rota em web é um “caminho” que será “chamado” por uma aplicação

- Este caminho é responsável por um ou mais serviços.

Cada rota pode ter uma ou mais funções, e ela deve ser única na aplicação web, ao receber uma chamada ela faz todo o processamento necessário para retornar os dados que foram solicitados.

Roteamento

- O Roteamento refere-se à determinação de como um aplicativo responde a uma solicitação do cliente por um endpoint específico, que é uma URI (ou caminho) e um método de solicitação HTTP específico (GET, POST, etc).

- Cada rota pode ter uma ou mais funções de manipulação, que são executadas quando a rota é correspondida.

Exemplo

- Url: http://localhost:8080/mostrarClientes
  - Rota: /mostrarClientes
  - Função: Retornar uma tabela com todos os clientes cadastrados.
- Url: http://localhost:8080/calcularMedia
  - Rota: /calcularMedia
  - Função: Recebe duas notas e retorna a média das notas.

Arquivo *.htaccess*

- O arquivo .htaccess é um arquivo de configuração do Apache.
- Seu conteúdo dará instruções ao Apache para que o servidor se comporte de uma determinada maneira.
- Pode ser utilizado para o tratamento e manipulação de rotas em php.

Redireciona todas as rotas para: index.php

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ /index.php
```

C:\xampp\htdocs\.htaccess

Arquivo: .htaccess:

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ /index.php
```

- O arquivo ***.htaccess*** fará o redirecionamento de qualquer requisição para o arquivo index.php. O arquivo index.php será um roteador.
- http://localhost/teste
- http://localhost/teste/tesete2
- http://localhost/teste/n
- Todas as rotas acima serão processadas pelo arquivo index.php

Variável: $_SERVER['REQUEST_URI']

$_SERVER['REQUEST_URI'] => Essa variável é responsável por armazenar a URL digitada pelo usuário.

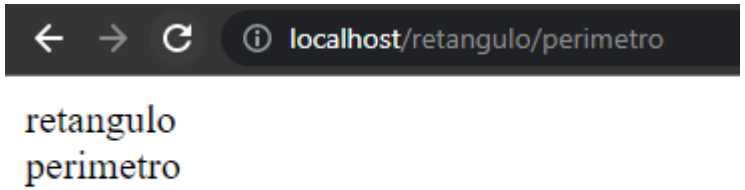
http://localhost/teste/n	• $_SERVER['REQUEST_URI']= /teste/n
http://localhost/retangulo/area	• $_SERVER['REQUEST_URI']= /retangulo/area
http://localhost/retangulo/perimetro	• $_SERVER['REQUEST_URI']= /retangulo/perimetro

Index.php

Observe a url digitada no navegador:
http://localhost/retângulo/perimetro

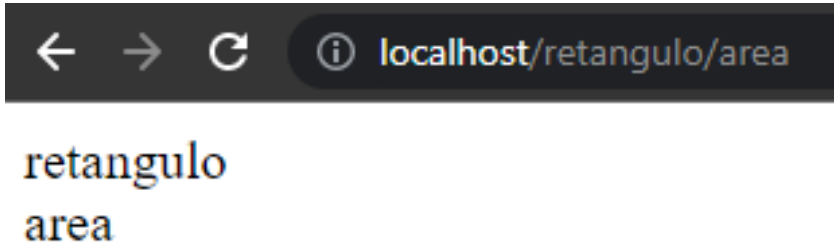
Observe a explosão da URL e as posições dos vetores

```
<?php
   $partesRota = explode("/", $_SERVER['REQUEST_URI']);
   echo  $partesRota[1]."<br>";
   echo  $partesRota[2];
?>
```



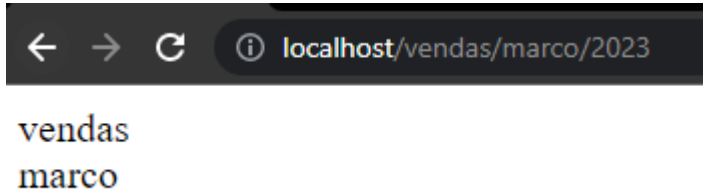
Index.php

```
//Testar com a rota : http://localhost/retangulo/perimetro
<?php
   $partesRota = explode("/", $_SERVER['REQUEST_URI']);
   echo  $partesRota[1]."<br>";
   echo  $partesRota[2];
?>
```



Qual o valor da posição 3 do vetor?

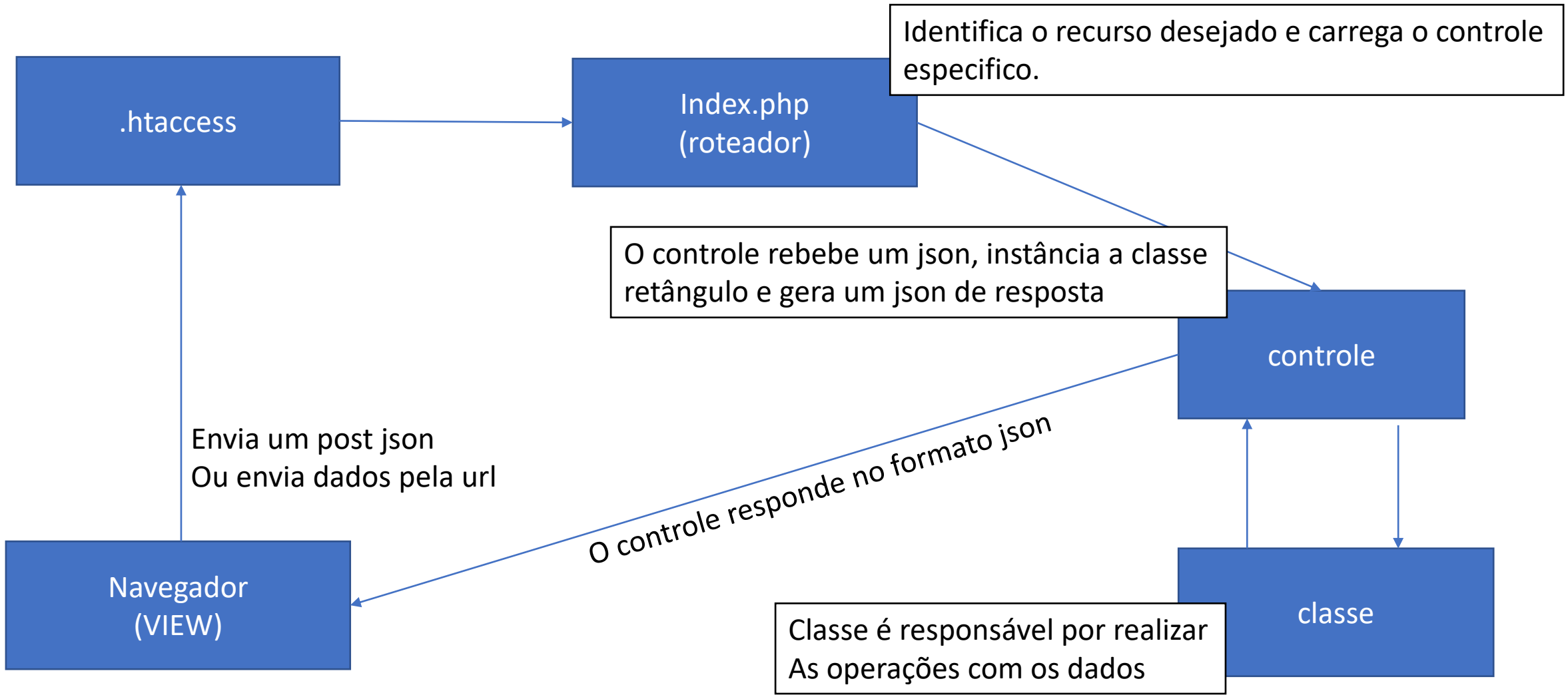
```php
<?php

    $partesRota = explode("/", $_SERVER['REQUEST_URI']);
    echo  $partesRota[1]."<br>";
    echo  $partesRota[2];

?>
```



Exemplo 1: API retângulo

- Será apresentada a criação de uma API no Modelo MVC.
- O front-end envia um GET com a URL no seguinte padrão:
  - http://localhost/retangulo/serviço/variavel1/variavel2
    - http://localhost/retangulo/area/3/4
    - http://localhost/retangulo/diagonal/3/4
    - http://localhost/retangulo/perimetro/3/4
- A api recebe do cliente dados e responde um JSON.

Arquitetura do sistema

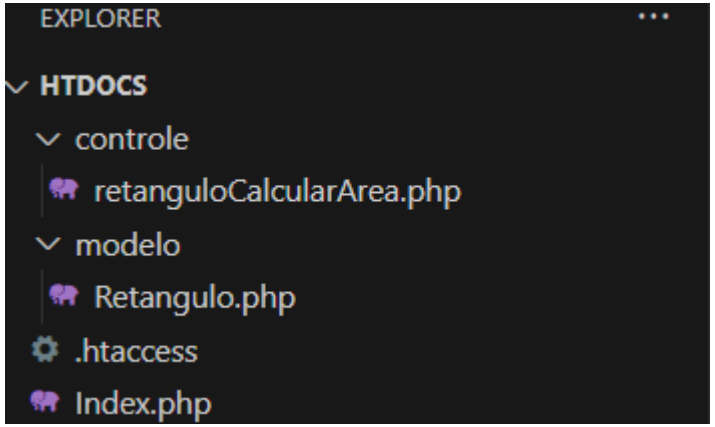


Exemplo: Arquivos

- Arquivos pasta raiz: \htdocs\
- Configuração e roteamento
  - .htaccess
  - Index.php
- Pasta: controle
  - retanguloCalcularArea.php
- Pasta: modelo
  - Retangulo.php

Arquivos

- Para fins didáticos apague todos os arquivos da pasta htdocs e posicione todos os arquivos no diretório: C:\xampp\htdocs
- Crie o arquivo .htaccess
- Crie o arquivo index.php
- Crie a pasta controle.
  - retanguloCalcularArea.php
- Crie a pasta modelo.
  - retanguloCalcularArea.php



Roteamento

- .htaccess
- Index.php

.htaccess

redireciona todas as requisições para o arquivo index.php

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ /index.php
```

Não encontrou o arquivo redireciona para **index.php**

Não encontrou o Diretório redireciona para **index.php**

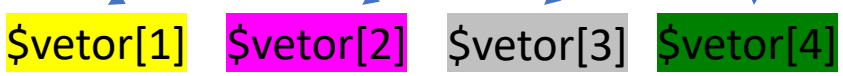
Index.php

```php
<?php
$vetor = explode("/", $_SERVER['REQUEST_URI']); // quebra a url em um array
$metodo = $_SERVER['REQUEST_METHOD']; //recupera o método de envido
if ($metodo == "GET") { //Verificar se o método de envio é get
    if ($vetor[1] == "retangulo") {
        if ($vetor[2] == "area") {
            require_once "controle/retanguloCalcularArea.php";
        } elseif ($vetor[2] == "diagonal") {
            require_once "controle/retanguloCalcularDiagonal.php";
        } elseif ($vetor[2] == "perimetro") {
            require_once "controle/retanguloCalcularPerimetro.php";
        } else {
            header("HTTP/1.1 404 Not Found");
        }
    } else if ($vetor[1] == "??") {
        //outra rota
    }
} else {
    header("HTTP/1.1 404 Not Found");
}
?>
```

http://localhost/retangulo/area/3/4

http://localhost/retangulo/diagonal/3/4

http://localhost/retangulo/perimetro/3/4

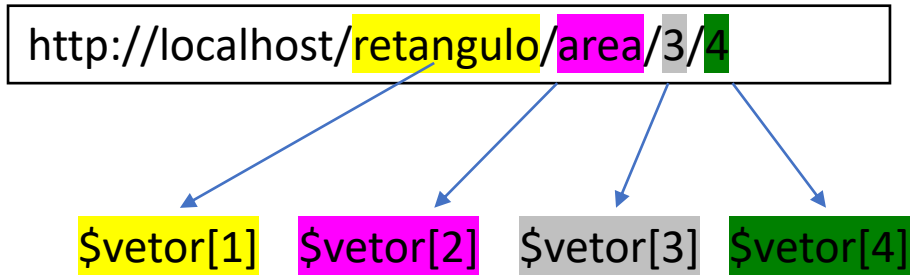


Controle

- retanguloCalcularArea.php
- retanguloCalcularDiagonal.php
- retanguloCalcularPerimetro.php

retanguloCalcularArea.php

```php
<?php
require_once "modelo/Retangulo.php";

$vetor = explode("/", $_SERVER['REQUEST_URI']);
$base      =   $vetor[3];
$altura    =   $vetor[4];

$r1 = new Retangulo();
$r1->setBase($base);
$r1->setAltura($altura);

$area = $r1->calcularArea();

$resposta = array();
$resposta['area'] = $area;

header("HTTP/1.1 200 OK");
header('Content-Type: application/json; charset=utf-8');
echo json_encode($resposta);
?>
```



Cria um array associativo contendo a resposta

Converte o array para o formato JSON

retanguloCalcularDiagonal.php

```php
<?php
require_once "modelo/Retangulo.php";

$vetor = explode("/", $_SERVER['REQUEST_URI']);
$base      =   $vetor[3];
$altura    =   $vetor[4];

$r1 = new Retangulo();
$r1->setBase($base);
$r1->setAltura($altura);

$perimetro = $r1->calcularDiagonal();

$resposta = array();
$resposta[diagonal'] = $perimetro;

header("HTTP/1.1 200 OK");
header('Content-Type: application/json; charset=utf-8');
echo json_encode($resposta);
?>
```

http://localhost/retangulo/diagonal/3/4

$vetor[1] $vetor[2] $vetor[3] $vetor[4]

Cria um array associativo contendo a resposta

Converte o array para o formato JSON

retanguloCalcularPerimetro.php

```php
<?php
require_once "modelo/Retangulo.php";

$vetor = explode("/", $_SERVER['REQUEST_URI']);
$base      =   $vetor[3];
$altura    =   $vetor[4];


$r1 = new Retangulo();
$r1->setBase($base);
$r1->setAltura($altura);

$perimetro = $r1->calcularPerimetro();

$resposta = array();
$resposta['perimetro'] = $perimetro;
header("HTTP/1.1 200 OK");

header('Content-Type: application/json; charset=utf-8');
echo json_encode($resposta);
?>
```

http://localhost/retangulo/diagonal/3/4

$vetor[1] $vetor[2] $vetor[3] $vetor[4]

Cria um array associativo contendo a resposta

Converte o array para o formato JSON

Modelo

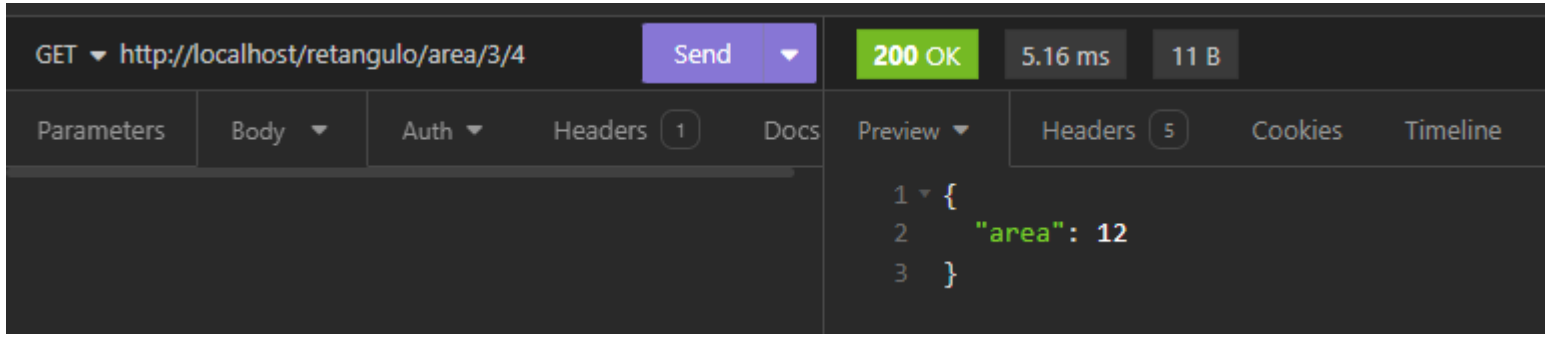
- Retangulo.php

Retangulo.php

```php
<?php
class Retangulo{
    private $base;
    private $altura;
    public function calcularArea()    {
        return ($this->altura * $this->base);
    }
    public function calcularDiagonal()  {
        return sqrt(pow($this->altura, 2) + pow($this->base, 2));
    }
    public function calcularPerimetro()    {
        return ($this->altura * 2 + $this->base * 2);
    }
    public function setBase($novaBase)    {
        $this->base = $novaBase;
    }
    public function getBase()    {
        return $this->base;
    }
    public function setAltura($novaAltura)    {
        $this->altura = $novaAltura;
    }
    public function getAltura()    {
        return $this->altura;
    }
}
?>
```

Teste de API: GET

- Observe que a rota não envia parâmetros via get
  - http://localhost/retangulo/area/base/altura
- Observe que base e a largura são enviadas em posições especificas na url. Na url o 3 é base do retângulo e 4 é a altura.



Enviar dados via POST

- Quando é utilizado o método post, os dados são enviados no corpo da requisição.
- Os dados tipicamente são enviados no formato json.
  - {"altura": "5" , "base": "10"}

```php
<?php
$vetor = explode("/", $_SERVER['REQUEST_URI']); // quebra a url em um array
$metodo = $_SERVER['REQUEST_METHOD']; //recupera o método de envido
if ($metodo == "POST") { //Verificar se o método de envio é get
    if ($vetor[1] == "retangulo") {
        if ($vetor[2] == "area") {
            require_once "controle/retanguloCalcularArea.php";
        } elseif ($vetor[2] == "diagonal") {
            require_once "controle/retanguloCalcularDiagonal.php";
        } elseif ($vetor[2] == "perimetro") {
            require_once "controle/retanguloCalcularPerimetro.php";
        } else {
            header("HTTP/1.1 404 Not Found");
        }
    } else if ($vetor[1] == "??") {
        //outra rota
    }
} else {
    header("HTTP/1.1 404 Not Found");
}
?>
```

É preciso mudar o index.php Observe que o método deve ser trocado para POST

retanguloCalcularArea.php

```php
<?php
require_once "modelo/Retangulo.php";
//recupera os dados enviados no corpo da requisição
$jsonRecebido = file_get_contents('php://input');
$obj = json_decode($jsonRecebido); //converte os dados em um objeto json
//recupera a propriedade base e a propriedade altura e base do objeto
$base = $obj->base;
$altura = $obj->altura;

$r1 = new Retangulo();
$r1->setBase($base);
$r1->setAltura($altura);
$area= $r1->calcularArea();

$resposta = array();
$resposta['area'] = $area;
header("HTTP/1.1 200 OK");
header('Content-Type: application/json; charset=utf-8');
echo json_encode($resposta);
?>
```

- file_get_contents('php://input');
  - Recupera os dados que foram enviado no corpo da requisição.
- $jsonRecebido é o texto json enviado no corpo da requisição.

retanguloCalcularDiagonal.php

```php
<?php
require_once "modelo/Retangulo.php";
//recupera os dados enviados no corpo da requisição
$jsonRecebido = file_get_contents('php://input');
$obj = json_decode($jsonRecebido); //converte os dados em um objeto json
//recupera a propriedade base e a propriedade altura e base do objeto
$base = $obj->base;
$altura = $obj->altura;

$r1 = new Retangulo();
$r1->setBase($base);
$r1->setAltura($altura);
$diagonal = $r1->calcularDiagonal();

$resposta = array();
$resposta['diagonal'] = $diagonal;
header("HTTP/1.1 200 OK");
header('Content-Type: application/json; charset=utf-8');
echo json_encode($resposta);
?>
```

- file_get_contents('php://input');
  - Recupera os dados que foram enviado no corpo da requisição.
- $jsonRecebido é o texto json enviado no corpo da requisição.

retanguloCalcularPerimetro.php

```php
<?php
require_once "modelo/Retangulo.php";
//recupera os dados enviados no corpo da requisição
$jsonRecebido = file_get_contents('php://input');
$obj = json_decode($jsonRecebido); //converte os dados em um objeto json
//recupera a propriedade base e a propriedade altura e base do objeto
$base = $obj->base;
$altura = $obj->altura;

$r1 = new Retangulo();
$r1->setBase($base);
$r1->setAltura($altura);
$perimetro= $r1->calcularPerimetro();

$resposta = array();
$resposta['perimetrol'] = $perimetro;
header("HTTP/1.1 200 OK");
header('Content-Type: application/json; charset=utf-8');
echo json_encode($resposta);
?>
```

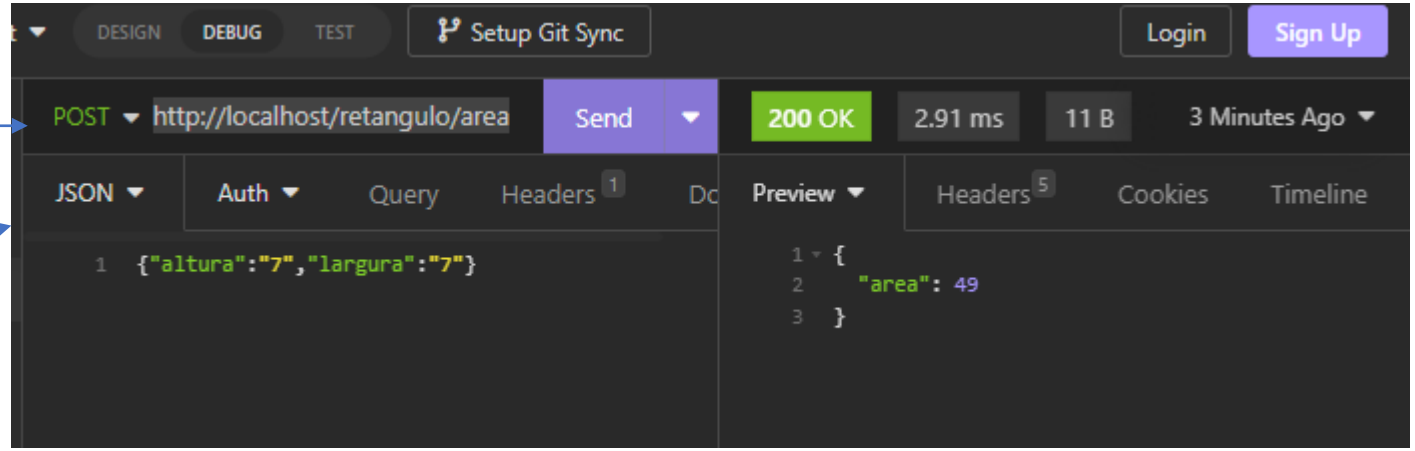
- file_get_contents('php://input');
  - Recupera os dados que foram enviado no corpo da requisição.
- $jsonRecebido é o texto json enviado no corpo da requisição.

Teste de API: POST

- Troque o método de envio para POST
- Troque o tipo de dado de envio para JSON

Troque o método de envio para post

Digite um json válido par testar

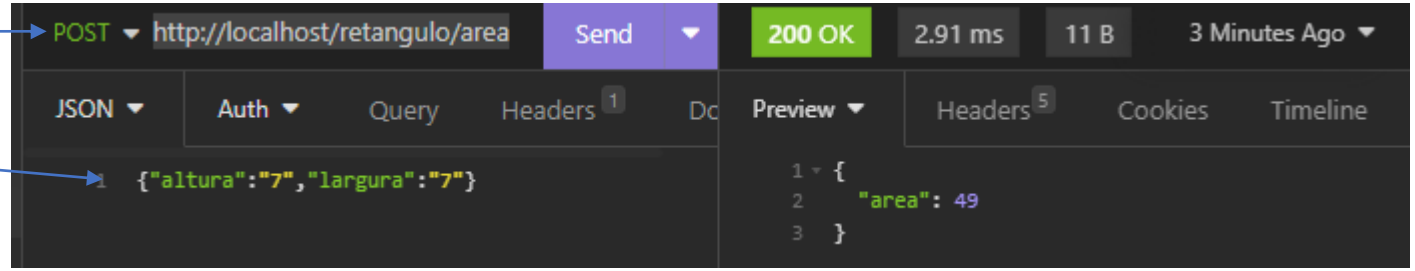


Teste de API

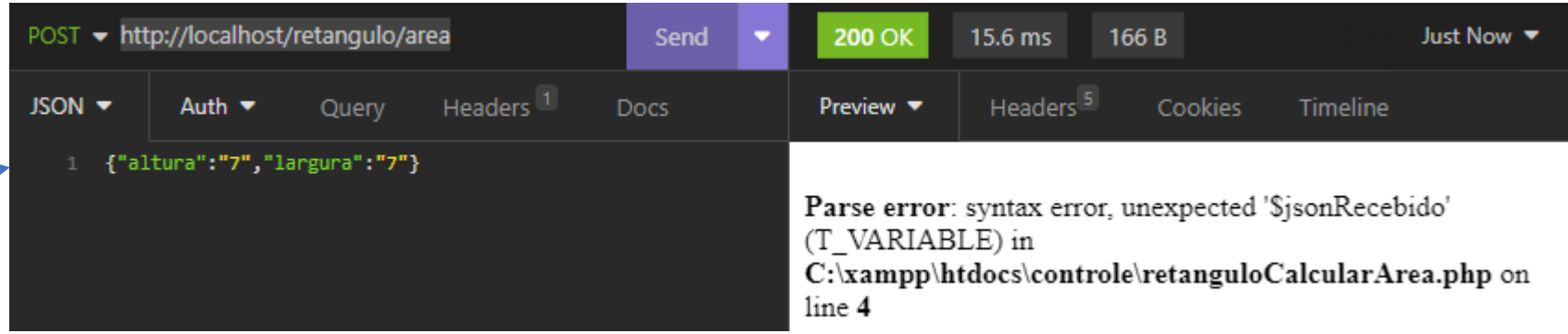
- Antes de construir o front-end é necessário testar a API já construída.
- É uma péssima prática construir tudo e testar no final.
  - A aplicação é complexa e encontrar problemas fica quase impossível

Troque o método de envio para post

Digite um json válido par testar



É fácil visualizar problemas e corrigir quando é utilizado o insomnia



Rotina de testes

- Antes de construir o front-end é necessário testar a API já construída.
- É uma péssima prática construir tudo e testar no final.
  - A aplicação é complexa e encontrar problemas fica quase impossível
- Apenas após funcionar no teste construa o front.