




SW-II API's

Prof. Anderson Vanin

Exemplo prático

Para demonstrar o funcionamento e a construção de uma API, utilizaremos a linguagem PHP para a construção da API e de uma pequena Aplicação Web que irá consumir esta API acessando algumas informações de um arquivo estático.

XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]



XAMPP Control Panel v3.3.0

Config

Netstat

Shell






Explorer

Services

Help

Quit

Modules

Service	Module	PID(s)	Port(s)	Actions			
	Apache	11976 23028	80, 443	Stop	Admin	Config	Logs
	MySQL	20080	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

09:34:26 [main] Starting Check-Timer

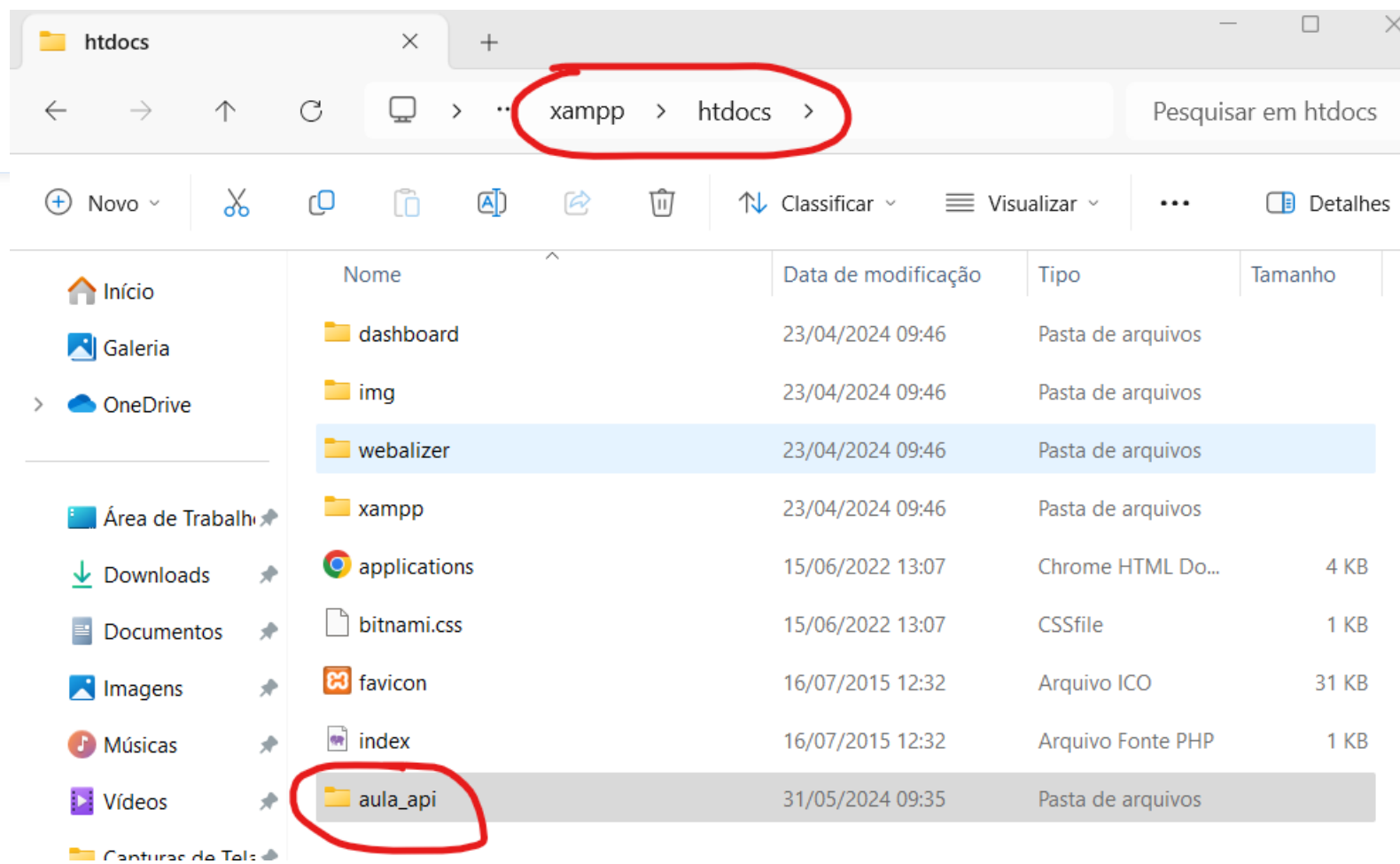
09:34:26 [main] Control Panel Ready

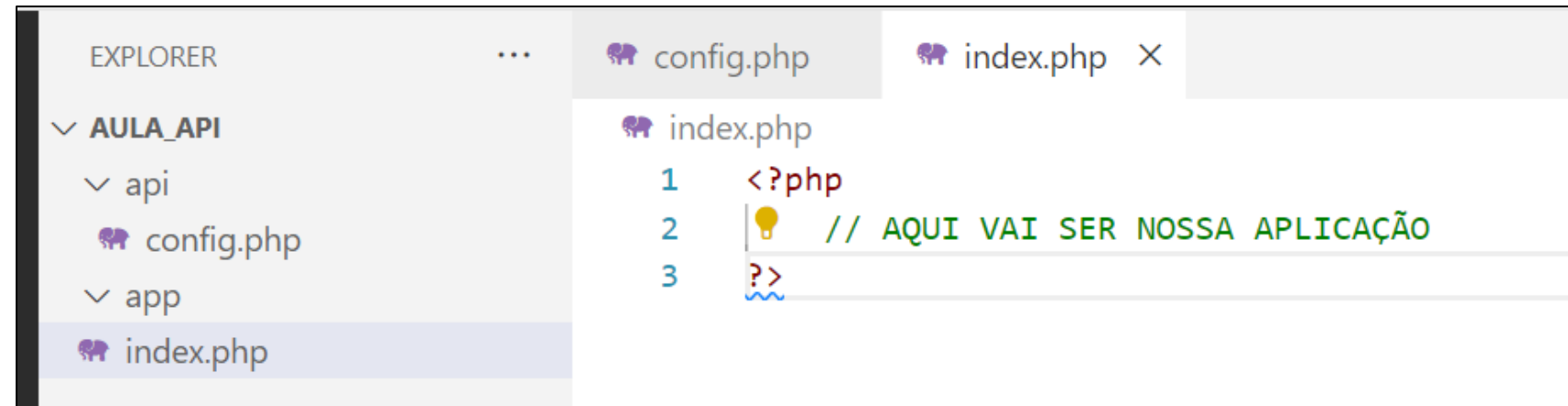
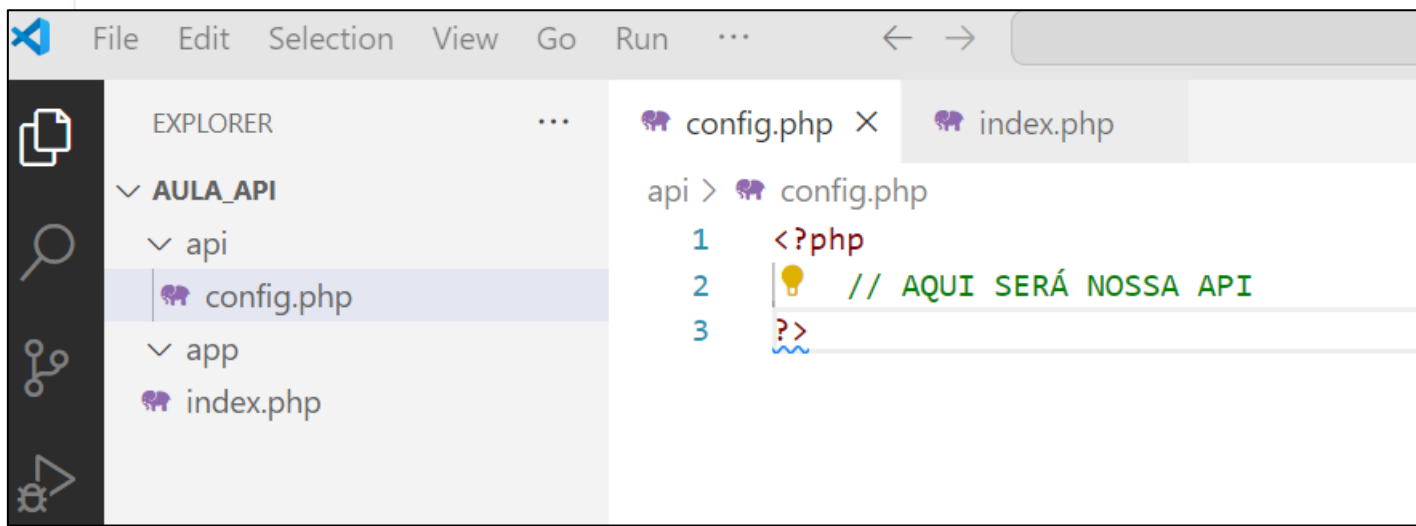
09:34:31 [Apache] Attempting to start Apache app...

09:34:31 [Apache] Status change detected: running

09:34:32 [mysql] Attempting to start MySQL app...

09:34:32 [mysql] Status change detected: running





config.php X

api > config.php

```
1  <?php
2      // AQUI SERÁ NOSSA API QUE TERÁ DUAS CONSTANTES:
3      // - API ESTÁ ATIVA OU NÃO
4      // - VERSÃO DA API
5  ?>
```

config.php X

api > config.php > ...

```
1  <?php
2      // AQUI SERÁ NOSSA API QUE TERÁ DUAS CONSTANTES:
3      // - API ESTÁ ATIVA OU NÃO
4      // - VERSÃO DA API
5
6      define('API_IS_ACTIVE',true);
7      define('API_VERSION','1.0.0');
8
```

EXPLORER

...

✓ AULA_API

✓ api

🐘 config.php

🐘 response.php

✓ app

🐘 index.php

🐘 response.php ✕

api > 🐘 response.php > 🔗 Response > 📦 resposta()

1 <?php

2

0 references | 0 implementations

3 class Response

4 {

5 | //Aqui vamos criar um método que dará a resposta da API

0 references | 0 overrides

6 | public static function resposta(\$status = 200,\$message='success',\$data=null){

7 | | //Aqui montamos o corpo da resposta

8 | }

9 }

response.php X

api > response.php > Response

1 <?php

2

0 references | 0 implementations

3 class Response

4 {

5 //Aqui vamos criar um método que dará a resposta da API

0 references | 0 overrides

6 public static function resposta(\$status = 200,\$message='success',\$data=null)

7 {

8 //Aqui montamos o corpo da resposta

9

10 // Configuramos o tipo de resposta (JSON)

11 header('Content-Type: application/json');

12

13 // retorno da função

14 return json_encode([

15 'status' => \$status,

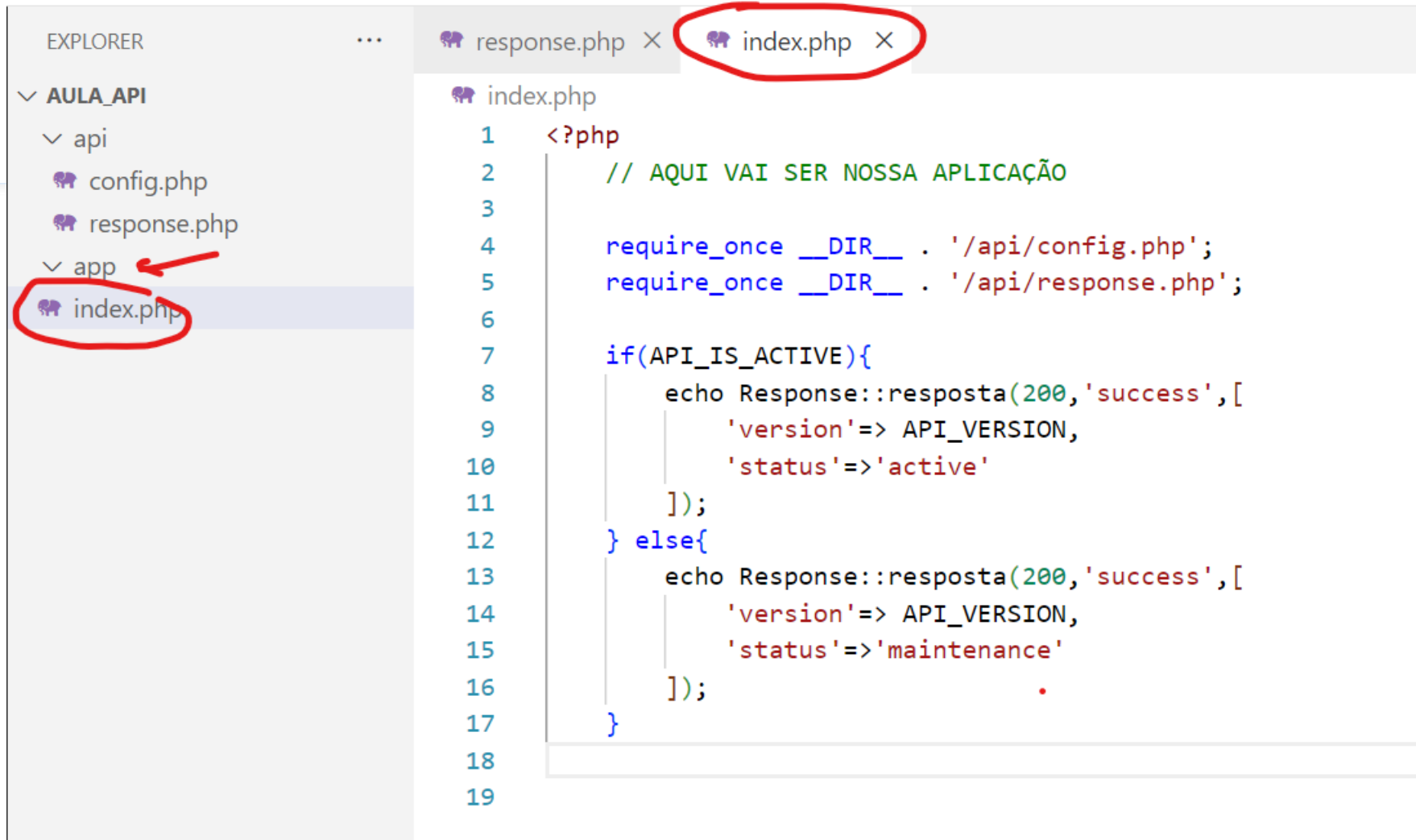
16 'mensagem' => \$message,

17 'dados' => \$data

18]);

19 }

20 }



EXPLORER

... response.php × index.php ×

▼ AULA_API

▼ api

- config.php
- response.php

▼ app

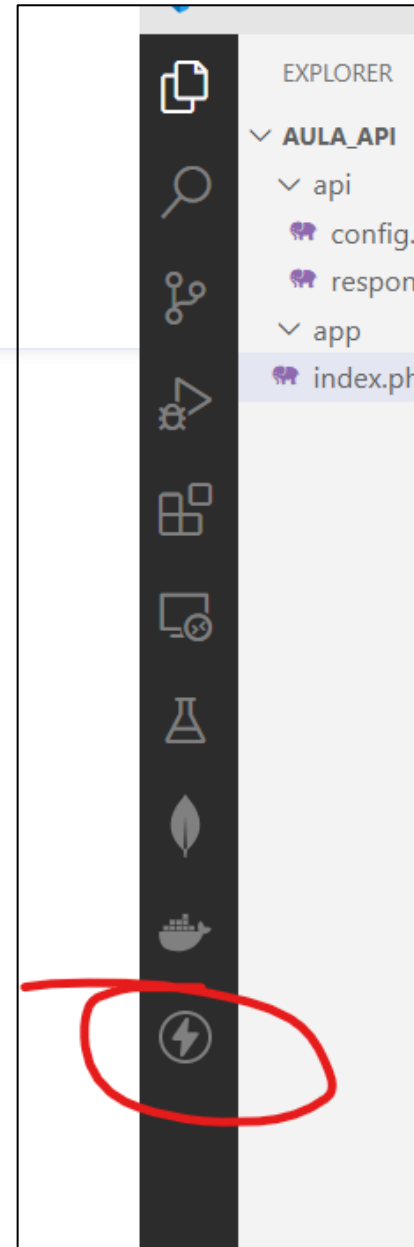
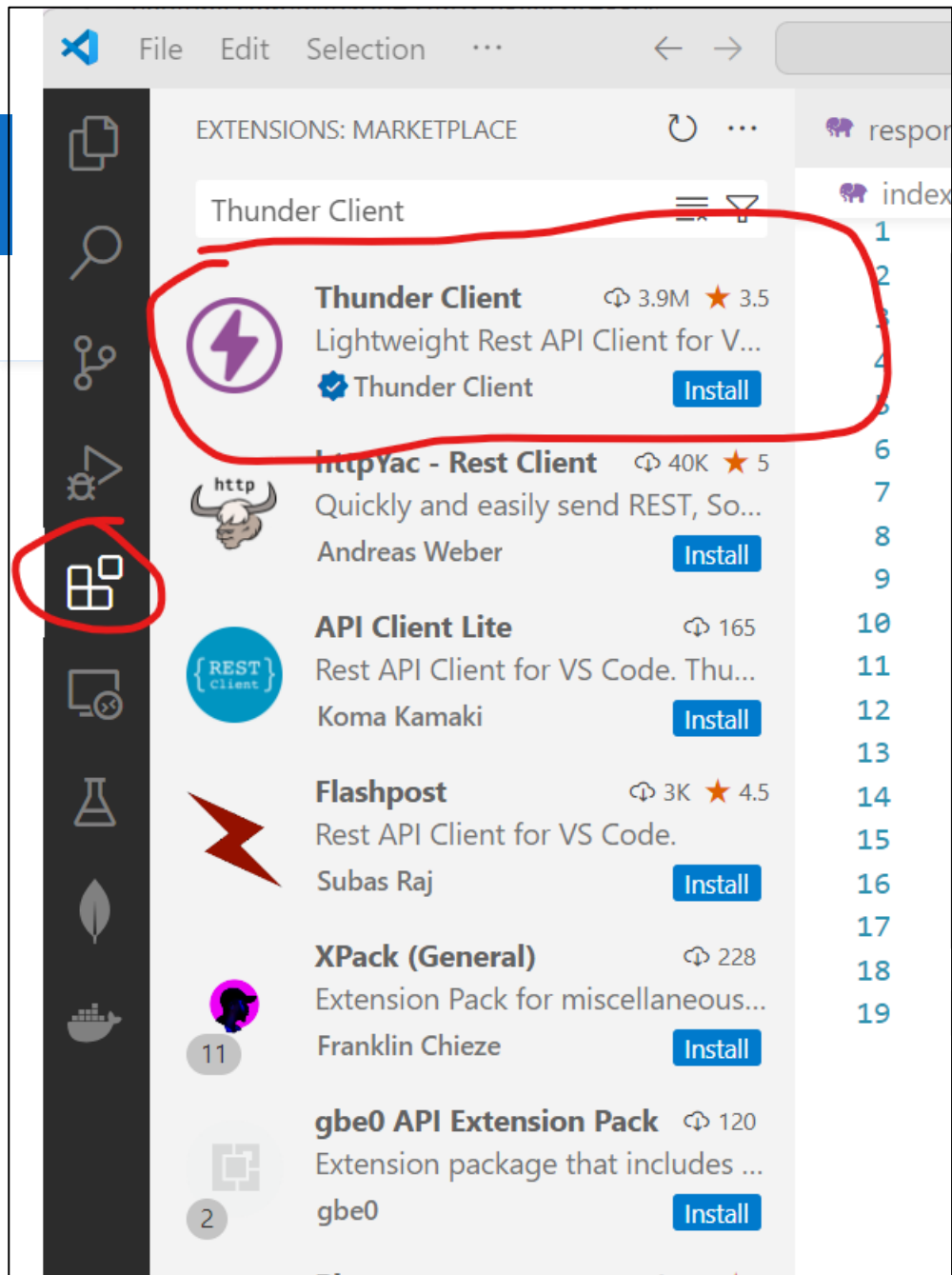
- index.php

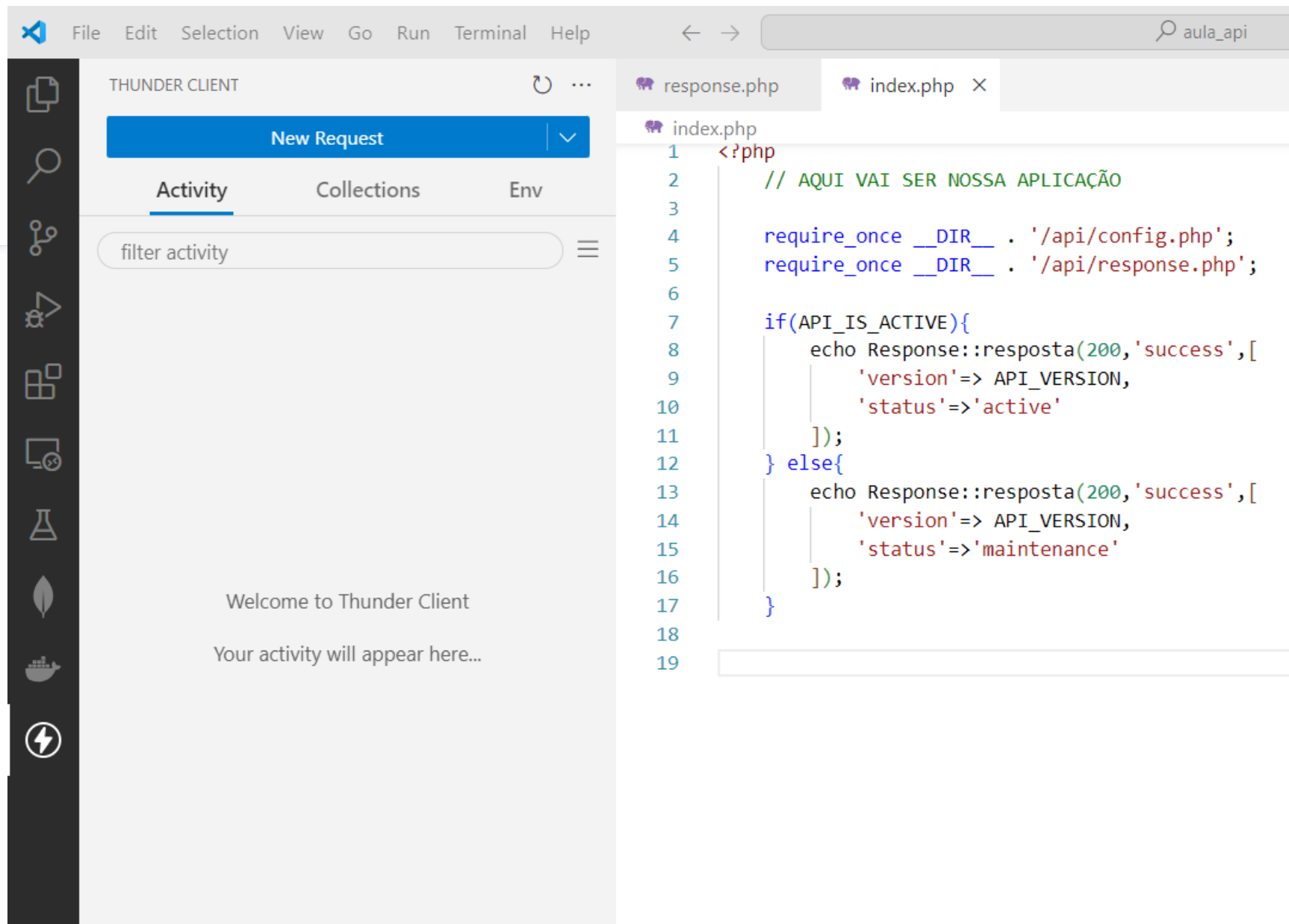
```
1 <?php
2 // AQUI VAI SER NOSSA APLICAÇÃO
3
4 require_once __DIR__ . '/api/config.php';
5 require_once __DIR__ . '/api/response.php';
6
7 if(API_IS_ACTIVE){
8     echo Response::resposta(200,'success',[
9         'version'=> API_VERSION,
10        'status'=>'active'
11    ]);
12 } else{
13     echo Response::resposta(200,'success',[
14         'version'=> API_VERSION,
15         'status'=>'maintenance'
16     ]);
17 }
18
19
```

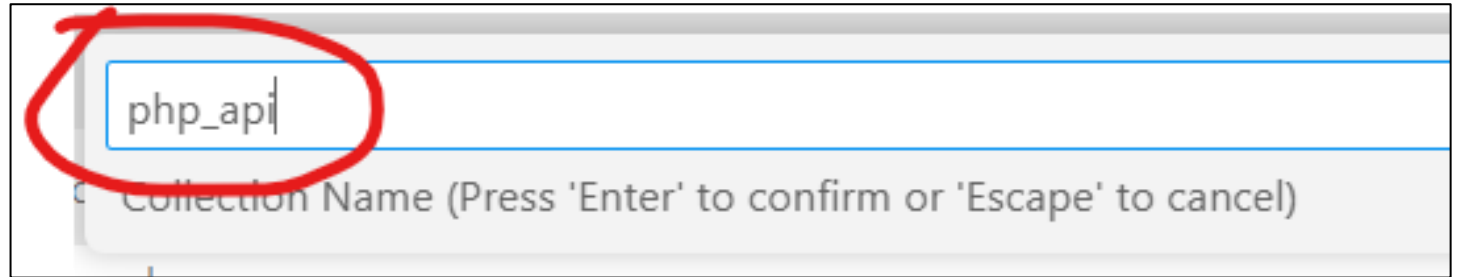
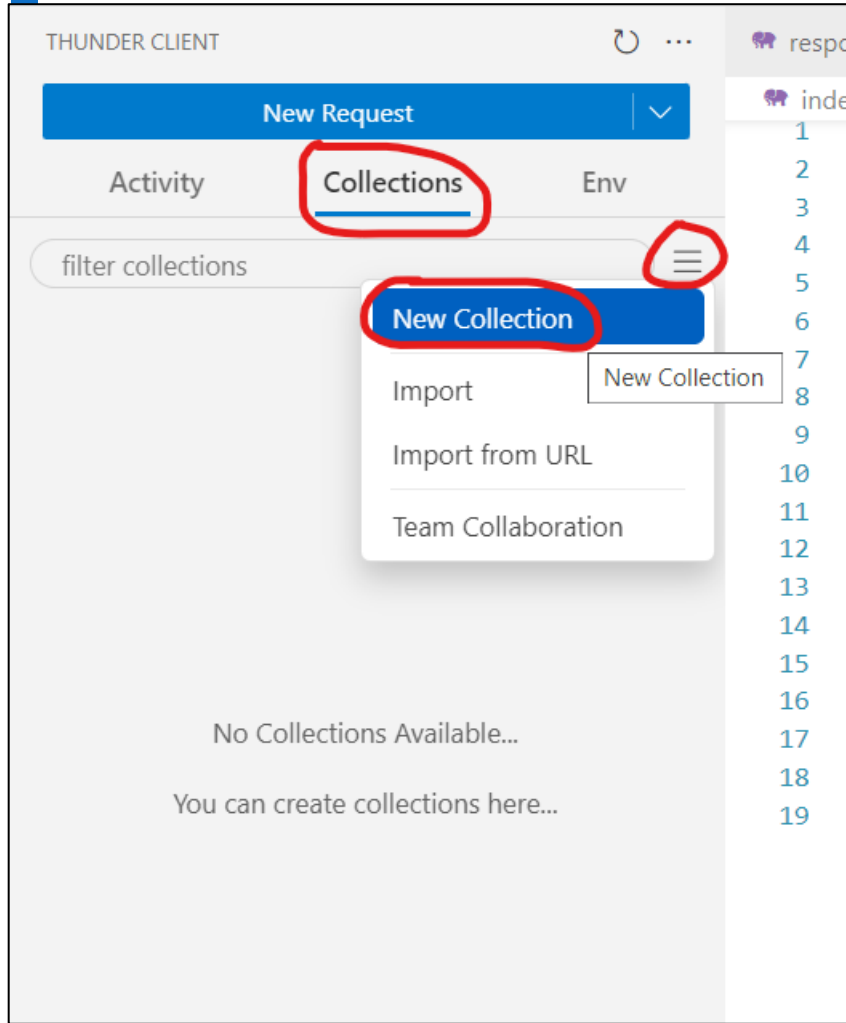


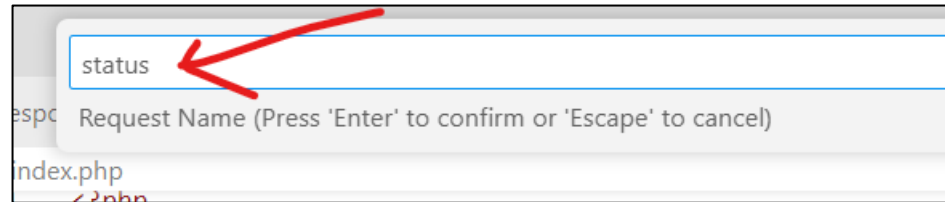
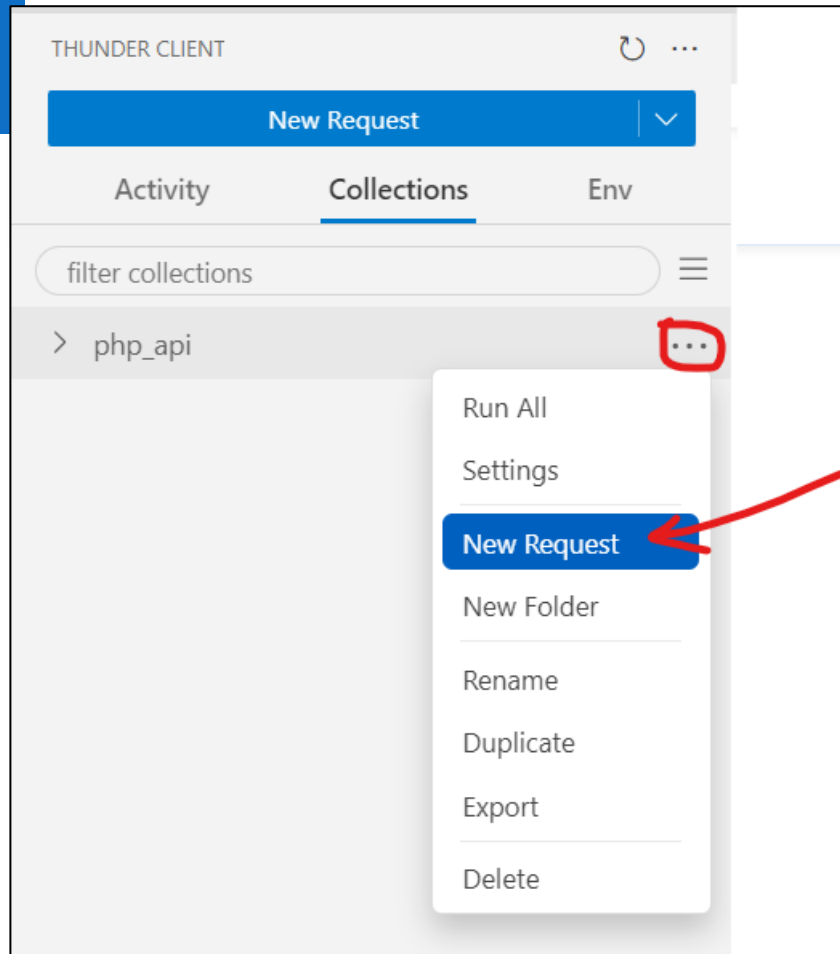
Ferramentas para API

- Postman
- Insomnia
- Swagger









Env

—
—
—

just now

Send

Pre Run

value

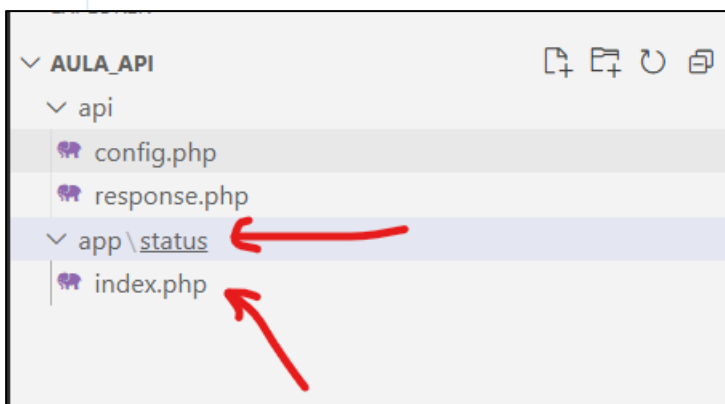
Time:

[Docs](#)
$$\{ \} =$$

Ctrl E

Pequena alteração

- Na pasta app onde fica a nossa aplicação, vamos dividir por partes. Como nesse primeiro exemplo queremos somente ver o status de nossa api, criamos uma pasta chamada status e dentro dela o arquivo index.php que criamos anteriormente.



```
app > status > index.php
1  <?php
2      // AQUI VAI SER NOSSA APLICAÇÃO
3
4      require_once __DIR__ . '/../api/config.php';
5      require_once __DIR__ . '/../api/response.php';
6
7      if(API_IS_ACTIVE){
8          echo Response::resposta(200,'success',[
9              'version'=> API_VERSION,
10             'status'=>'active'
11         ]);
12     } else{
13         echo Response::resposta(200,'success',[
14             'version'=> API_VERSION,
15             'status'=>'maintenance'
16         ]);
17     }
18
19
```

Nova url de requisição



Endpoint

- Este será um dos nossos endpoints de nossa API:

http://localhost/aula_api/app/status/

THUNDER CLIENT



response.php

index.php

TC status



New Request

Activity

Collections

Env

filter collections

php_api

GET status

just now

GET

http://localhost/aula_api/app/status/

Send

Query

Headers 2

Auth

Body

Tests

Pre Run

Query Parameters



parameter

value

Status: 200 OK Size: 81 Bytes Time: 8 ms

Response

Headers 6

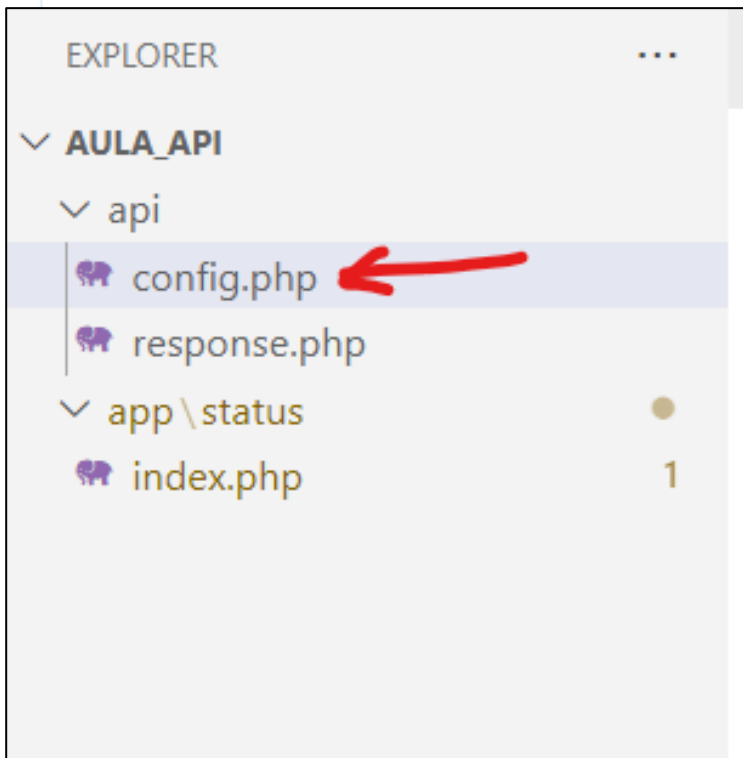
Cookies

Results

Docs



```
1 {
2   "status": 200,
3   "mensagem": "success",
4   "dados": {
5     "version": "1.0.0",
6     "status": "active"
7   }
8 }
```



```
api >  config.php > ...  
1  <?php  
2      // AQUI SERÁ NOSSA API QUE TERÁ DUAS CONSTANTES:  
3      // - API ESTÁ ATIVA OU NÃO  
4      // - VERSÃO DA API  
5  
6      define('API_IS_ACTIVE', false);  
7      define('API_VERSION', '1.0.0');  
8
```

response.php config.php index.php 1 TC status X

GET

http://localhost/aula_api/app/status/

Send

Query

Headers 2

Auth

Body

Tests

Pre Run

Query Parameters

☐

parameter

value

Status: 200 OK Size: 86 Bytes Time: 20 ms

Response

Headers 6

Cookies

Results

Docs



```
1 {  
2   "status": 200,  
3   "mensagem": "success",  
4   "dados": {  
5     "version": "1.0.0",  
6     "status": "maintenance"  
7   }  
8 }
```




Voltando a API



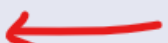

- Vamos criar um arquivo que representará a nossa base de dados.

EXPLORER


...  response.php  config.php


▼ **AULA_API**

▼ **api** 

-  config.php
-  data.php 
-  response.php

▼ app \ status ●

-  index.php 1

api >  data.php

1

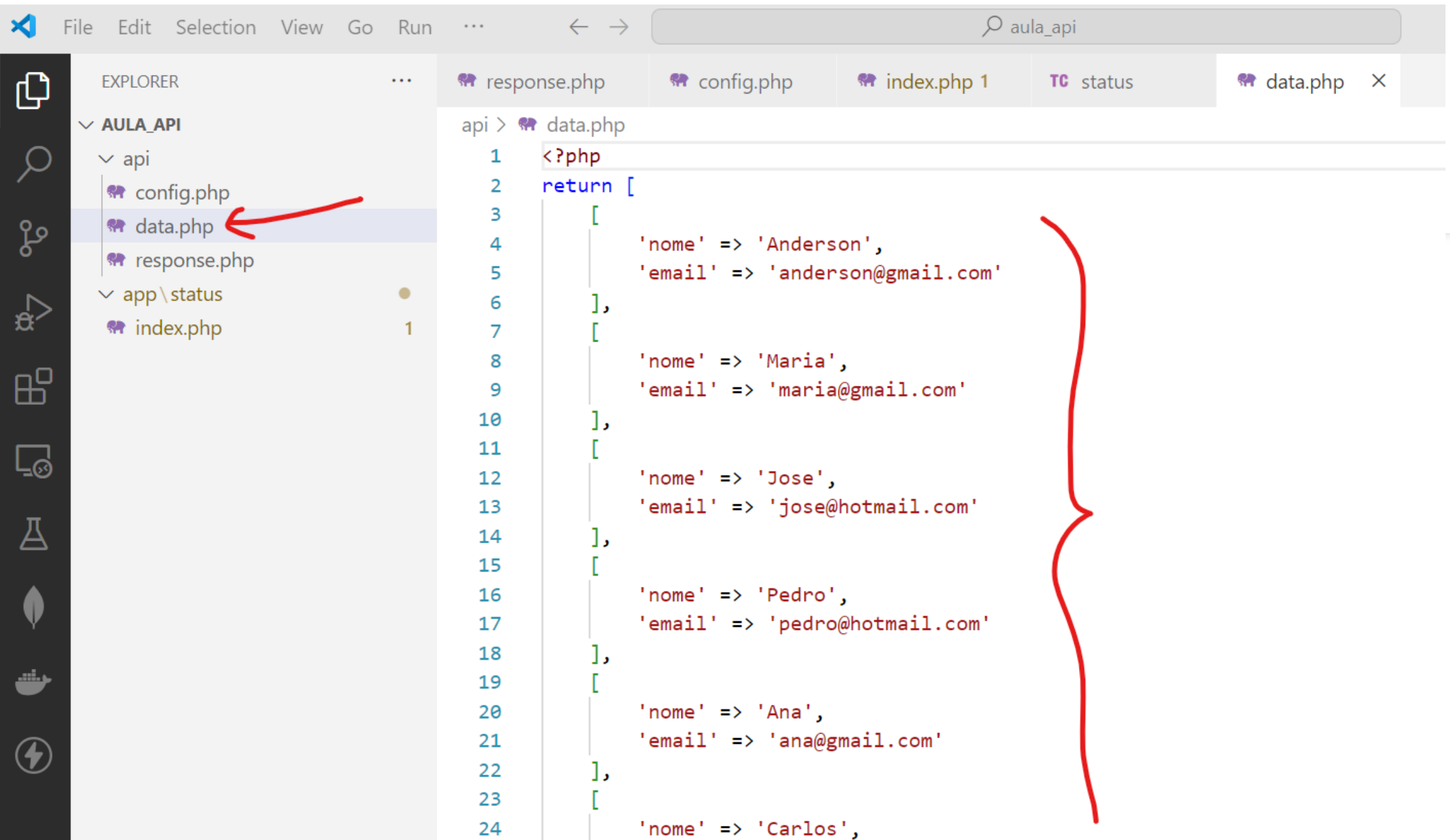
File Edit Selection View Go Run ... aula_api

EXPLORER

- ▼ AULA_API
 - ▼ api
 - config.php
 - data.php
 - response.php
 - ▼ app\status
 - index.php

api > data.php

```
1 <?php
2 return [
3     [
4         'nome' => 'Anderson',
5         'email' => 'anderson@gmail.com'
6     ],
7     [
8         'nome' => 'Maria',
9         'email' => 'maria@gmail.com'
10    ],
11    [
12        'nome' => 'Jose',
13        'email' => 'jose@hotmail.com'
14    ],
15    [
16        'nome' => 'Pedro',
17        'email' => 'pedro@hotmail.com'
18    ],
19    [
20        'nome' => 'Ana',
21        'email' => 'ana@gmail.com'
22    ],
23    [
24        'nome' => 'Carlos',
```



Voltando a API

Com base neste arquivo poderemos trazer algumas informações:

- Quantidade de Usuários
- Lista de domínios (gmail, Hotmail, etc..)

Para isso vamos primeiro fazer uma alteração no arquivo de resposta da api

api > response.php > Response > resposta()

1 <?php

2

2 references | 0 implementations

3 class Response

4 {

5 //Aqui vamos criar um método que dará a resposta da API

2 references | 0 overrides

6 public static function resposta(\$status = 200,\$message='success',\$data=null)

7 {

8 header('Content-Type: application/json');

9

10 // Checar se a API está ativa ou não

11 if(!API_IS_ACTIVE){

12 return json_encode([

13 'status' => 400,

14 'mensagem' => 'Api is not running',

15 'api_version' => API_VERSION,

16 'time_response' => time(),

17 'datetime_response' => date('Y-m-d H:i:s'),

18 'dados' => null

19], JSON_PRETTY_PRINT);

20 }

21

22 // retorno da função

23 return json_encode([

24 'status' => \$status,

25 'mensagem' => \$message,

26 'api_version' => API_VERSION,

27 'time_response' => time(),

28 'datetime_response' => date('Y-m-d H:i:s'),

29 'dados' => \$data

30], JSON_PRETTY_PRINT);

31 }

32 }

GET

Query Headers ² Auth Body Tests Pre Run

Query Parameters

<input type="checkbox"/>	parameter	value
--------------------------	-----------	-------

Status: 200 OK Size: 234 Bytes Time: 46 ms

Response Headers ⁶ Cookies Results Docs

```
1  {
2    "status": 200,
3    "mensagem": "success",
4    "api_version": "1.0.0",
5    "time_response": 1717942488,
6    "datetime_response": "2024-06-09 16:14:48",
7    "dados": {
8      "version": "1.0.0",
9      "status": "active"
10   }
11 }
```

EXPLORER

... response.php config.php index.php X TC status data.php

▼ AULA_API

▼ api

config.php

data.php

response.php

▼ app\status

index.php ←

app > status > index.php

```
1 <?php
2 // AQUI VAI SER NOSSA APLICAÇÃO
3
4 require_once __DIR__ . '/../api/config.php';
5 require_once __DIR__ . '/../api/response.php';
6
7 // if(API_IS_ACTIVE){
8 //     echo Response::resposta(200,'success',[
9 //         'version'=> API_VERSION,
10 //         'status'=>'active'
11 //     ]);
12 // } else{
13 //     echo Response::resposta(200,'success',[
14 //         'version'=> API_VERSION,
15 //         'status'=>'maintenance'
16 //     ]);
17 // }
18
19 echo Response::resposta(200,'success',[
20     'version'=> API_VERSION,
21     'status'=>'maintenance'
22 ]);
```

response.php Xconfig.phpindex.phpTC status Xdata.php

GET http://localhost/aula_api/app/status/ Send

QueryHeaders 2AuthBodyTestsPre Run

Query Parameters

☐

parameter

value

Status: 200 OKSize: 239 BytesTime: 20 ms

ResponseHeaders 6CookiesResultsDocs

```
1 {
2   "status": 200,
3   "mensagem": "success",
4   "api_version": "1.0.0",
5   "time_response": 1717942590,
6   "datetime_response": "2024-06-09 16:16:30",
7   "dados": {
8     "version": "1.0.0",
9     "status": "maintenance"
10  }
11 }
```

▼ AULA_API

▼ api

🐘 config.php

🐘 data.php

🐘 response.php

▼ app\status

🐘 index.php

app > status > 🐘 index.php

```
1 <?php
2 // AQUI VAI SER NOSSA APLICAÇÃO
3
4 require_once __DIR__ . '/../../api/config.php';
5 require_once __DIR__ . '/../../api/response.php';
6
7 // if(API_IS_ACTIVE){
8 //     echo Response::resposta(200,'success',[
9 //         'version'=> API_VERSION,
10 //         'status'=>'active'
11 //     ]);
12 // } else{
13 //     echo Response::resposta(200,'success',[
14 //         'version'=> API_VERSION,
15 //         'status'=>'maintenance'
16 //     ]);
17 // }
18
19 // echo Response::resposta(200,'success',[
20 //     'version'=> API_VERSION,
21 //     'status'=>'maintenance'
22 //     ]);
23
24 echo Response::resposta(200,'API is running!');
```




Agora vamos criar um outro endpoint

EXPLORER

... response.php config.php index.php ...\status TC status index.php ...\get_all_data X data.php

▼ AULA_API

▼ api

- config.php
- data.php
- response.php

▼ app

- ▼ get_all_data
 - index.php
- > status

app > get_all_data > index.php

```
1 <?php
2 // Este endpoint vai trazer todos os dados do arquivo data.php
3
4 require_once __DIR__ . '/../api/config.php';
5 require_once __DIR__ . '/../api/response.php';
6
7 $data = require_once __DIR__ . '/../api/data.php';
8
9
10 echo Response::resposta(200, 'sucess', $data);
```

THUNDER CLIENT

New Request

Activity Collections Env

filter collections

php_api

GET status 46 mins ago

Run All

Settings

New Request

New Folder

Rename

Duplicate

Export

Delete

get_dados

Request Name (Press 'Enter' to confirm or 'Escape' to cancel)

THUNDER CLIENT

New Request

Activity Collections Env

filter collections

php_api

GET status 46 mins ago

GET get_dados just now

GET http://localhost/aula_api/app/get_all_data/

Send

Query Headers Auth Body Tests Pre Run


Query Parameters

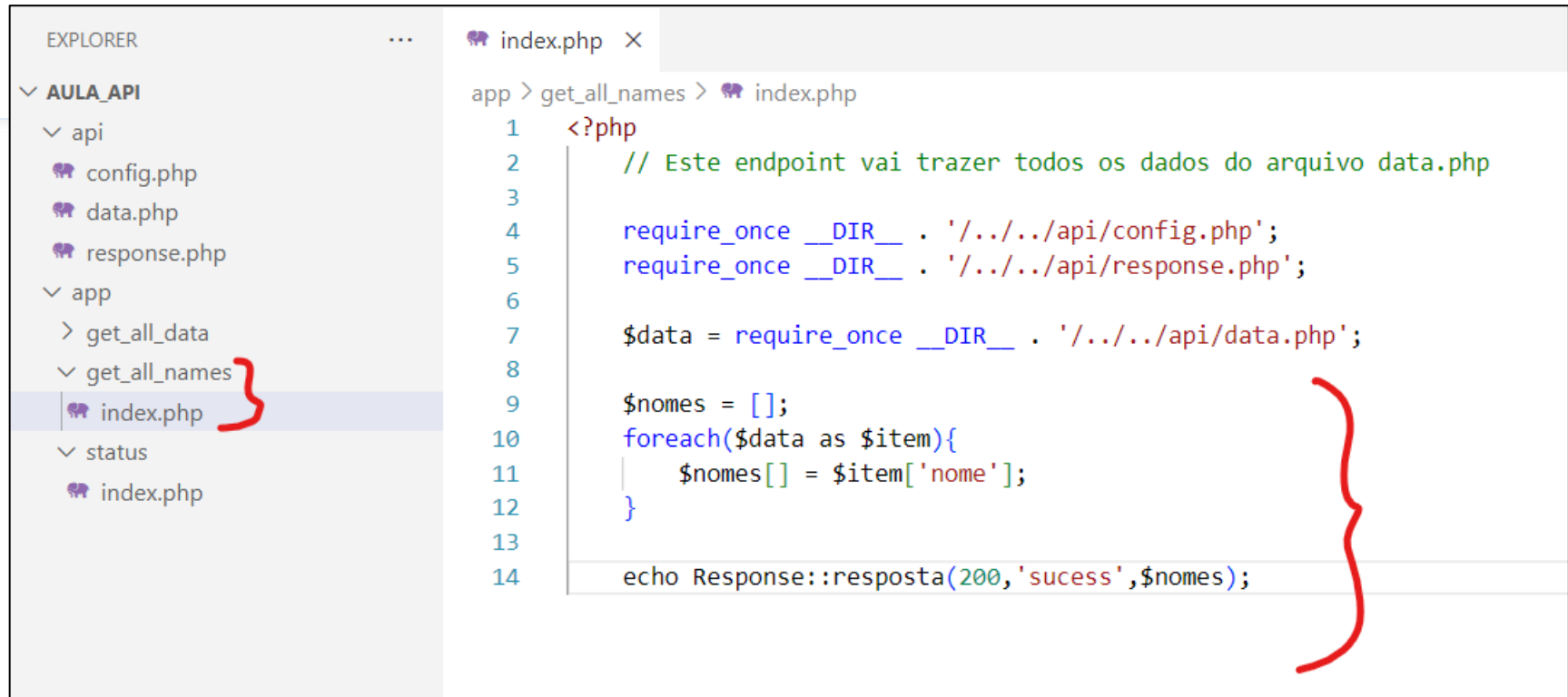
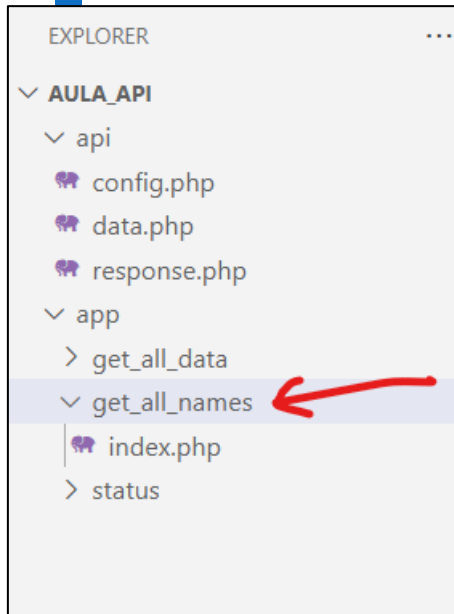
parameter	value
-----------	-------

Status: 200 OK Size: 1.04 KB Time: 31 ms

Response Headers Cookies Results Docs

```
1 {
2   "status": 200,
3   "mensagem": "sucess",
4   "api_version": "1.0.0",
5   "time_response": 1717943298,
6   "datetime_response": "2024-06-09 16:28:18",
7   "dados": [
8     {
9       "nome": "Anderson",
10      "email": "anderson@gmail.com"
11    },
12    {
13      "nome": "Maria",
14      "email": "maria@gmail.com"
15    },
16    {
17      "nome": "Jose",
18      "email": "jose@hotmail.com"
19    },
20    {
21      "nome": "Pedro",
22      "email": "pedro@hotmail.com"
23    },
24    {
25      "nome": "Ana",
26      "email": "ana@gmail.com"
27    },
28    {
29      "nome": "Carlos",
```

- 
- Os próximos endpoints que iremos criar, nos retornará os nomes das pessoas e o total de cadastros.



THUNDER CLIENT

New Request

Activity Collections Env

filter collections

php_api

- GET status
1 hour ago
- GET get_dados
15 mins ago
- GET get_nomes
just now

index.php TC get_nomes X

GET http://localhost/aula_api/app/get_all_names/ Send

Query Headers 2 Auth Body Tests Pre Run

Query Parameters

<input type="checkbox"/>	parameter	value
--------------------------	-----------	-------

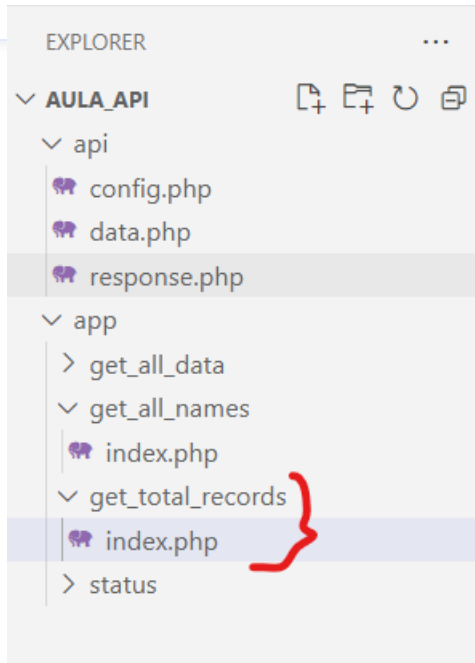
Status: 200 OK Size: 351 Bytes Time: 22 ms

Response Headers 6 Cookies Results Docs

```
1 {
2   "status": 200,
3   "mensagem": "sucess",
4   "api_version": "1.0.0",
5   "time_response": 1717944236,
6   "datetime_response": "2024-06-09 16:43:56",
7   "dados": [
8     "Anderson",
9     "Maria",
10    "Jose",
11    "Pedro",
12    "Ana",
13    "Carlos",
14    "Julia",
15    "Marcos",
16    "Juliana",
17    "Carla"
18  ]
19 }
```



Nova rota para retornar o total de registros

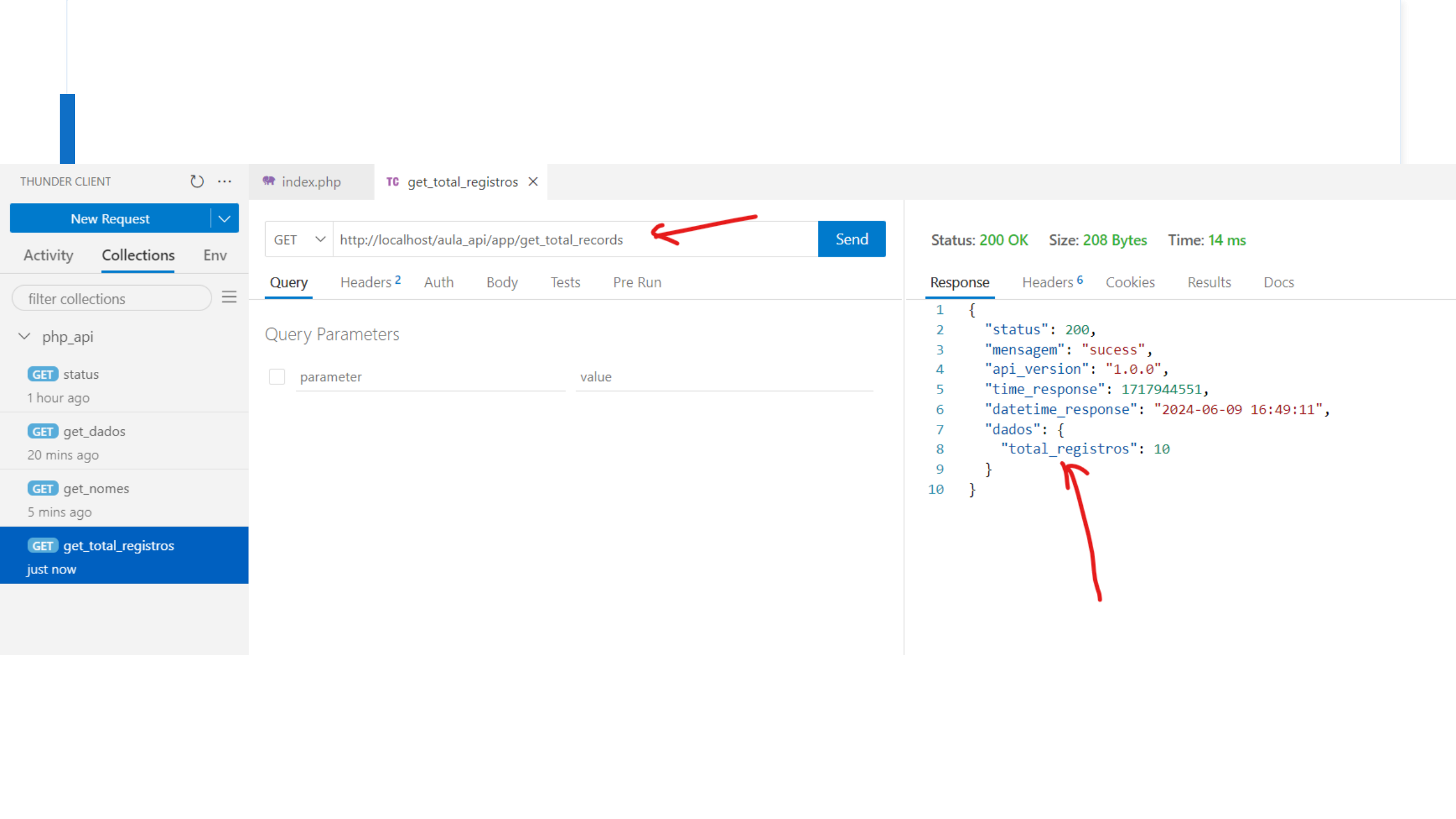


index.php X

app > get_total_records > index.php

```
1 <?php
2 // Este endpoint vai trazer todos os dados do arquivo data.php
3
4 require_once __DIR__ . '/../../api/config.php';
5 require_once __DIR__ . '/../../api/response.php';
6
7 $data = require_once __DIR__ . '/../../api/data.php';
8
9 echo Response::resposta(200,'sucess',['total_registros' => count($data)]);
```







Buscando por um cliente específico

EXPLORER

▼ AULA_API

▼ api

- config.php
- data.php
- response.php

▼ app

- > get_all_data
- > get_all_names
- > get_total_records
- ▼ get_um_cliente
 - index.php
- > status

index.php X TC get_um_cliente

app > get_um_cliente > index.php

```
1  <?php
2      // Este endpoint vai trazer todos os dados do arquivo data.php
3
4      require_once __DIR__ . '/../api/config.php';
5      require_once __DIR__ . '/../api/response.php';
6
7      $data = require_once __DIR__ . '/../api/data.php';
8
9      // verificar se na url veio alguma identificação do cliente
10     if(isset($_GET['id'])){
11         $id = $_GET['id'];
12
13     } else{
14         echo Response::resposta(400,'error','Necessário informar o id');
15         exit;
16     }
17
18     if($id < 0 || $id > count($data) - 1){
19         echo Response::resposta(400,'error','Cliente não encontrado');
20         exit;
21     }
22
23
24     echo Response::resposta(200,'sucess',$data[$id]);
```

index.php

TC get_um_cliente X

GET

http://localhost/aula_api/app/get_um_cliente/

Send

Query

Headers 2

Auth

Body

Tests

Pre Run

Query Parameters

☐

parameter

value

Status: 200 OK

Size: 201 Bytes

Time: 31 ms

Response

Headers 6

Cookies

Results

Docs

1 {

2 "status": 400,

3 "mensagem": "error",

4 "api_version": "1.0.0",

5 "time_response": 1717945741,

6 "datetime_response": "2024-06-09 17:09:01",

7 "dados": "Necessário informar o id"

8 }

index.php

TC get_um_cliente X

GET

http://localhost/aula_api/app/get_um_cliente?id=1

Send

Query

Headers 2

Auth

Body

Tests

Pre Run

Query Parameters

☒

id

1

☐

parameter

value

Status: 200 OK

Size: 238 Bytes

Time: 26 ms

Response

Headers 6

Cookies

Results

Docs

1 {

2 "status": 200,

3 "mensagem": "sucess",

4 "api_version": "1.0.0",

5 "time_response": 1717945928,

6 "datetime_response": "2024-06-09 17:12:08",

7 "dados": {

8 "nome": "Maria",

9 "email": "maria@gmail.com"

10 }

11 }

Lembre-se

```
api > data.php
1  <?php
2  return [
3      [
4          'nome' => 'Anderson',
5          'email' => 'anderson@gmail.com'
6      ],
7      [
8          'nome' => 'Maria',
9          'email' => 'maria@gmail.com'
10     ],
11     [
12         'nome' => 'Jose',
13         'email' => 'jose@hotmail.com'
14     ],
15     [
16         'nome' => 'Pedro',
17         'email' => 'pedro@hotmail.com'
18     ],
19     [
20         'nome' => 'Ana',
21         'email' => 'ana@gmail.com'
22     ],
23     [
24         'nome' => 'Carlos',
25         'email' => 'carlos@gmail.com'
26     ],
27     [
28         'nome' => 'Julia',
29         'email' => 'julia@gmail.com'
30     ],
31     [
32         'nome' => 'Marcos',
33         'email' => 'marcos@hotmail.com'
34     ],
35     [
36         'nome' => 'Juliana',
37         'email' => 'juliana@gmail.com'
38     ]
39 ]
```

Handwritten red annotations on the right side of the code:

- Red arrow pointing to line 5: \emptyset
- Red arrow pointing to line 9: 1
- Red arrow pointing to line 13: 2
- Red arrow pointing to line 17: 3
- Red arrow pointing to line 21: 4
- Red dots (vertical ellipsis) between lines 25 and 35.