

SW-II

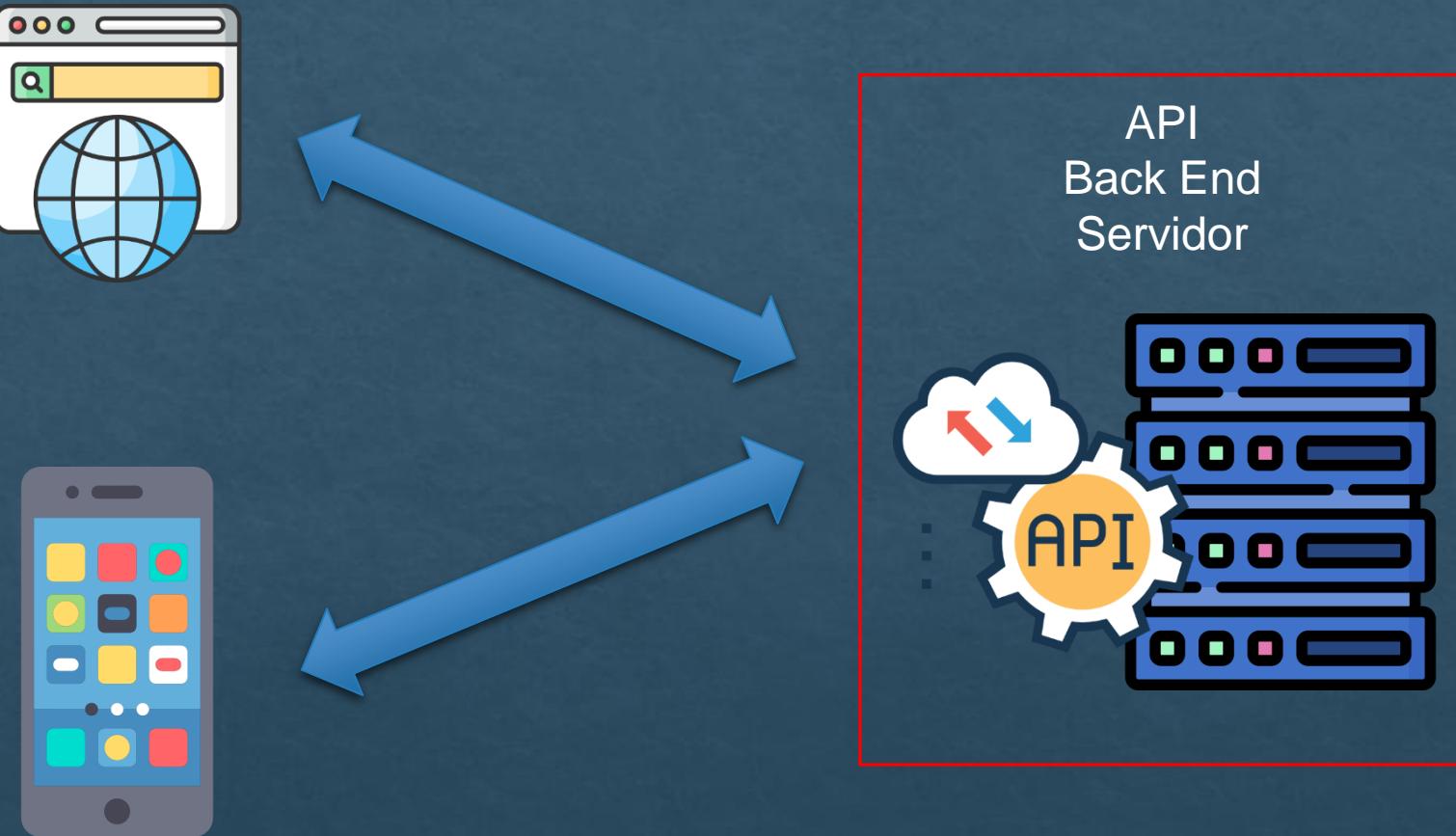
SISTEMAS WEB II

Prof. Anderson Vanin
AULA 14 – APIs com NODE e Banco de Dados

Conteúdos

- Conceitos de Back End e Front End
- Primeira API com node.js
- Conexão com Banco de Dados
- Query Params / Route Params / Body Params
- Métodos HTTP
- Status HTTP

Introdução



Introdução

Back End (Servidor)

```
{[  
  {  
    id: "456",  
    nome: "Fulano",  
    idade: 33,  
    email: fulano@email.com  
  },  
  {  
    id: "789",  
    nome: "Maria",  
    idade: 27,  
    email: maria@email.com  
  }  
]}
```

get - users
post - users
delete - users
put - users

```
[  
  {  
    id: "456",  
    nome: "Fulano",  
    idade: 33,  
    email:  
      fulano@email.com  
  },  
  {  
    id: "789",  
    nome: "Maria",  
    idade: 27,  
    email:  
      maria@email.com  
  }  
]
```

RESPOSTA

servidor.com/usuarios

REQUISIÇÃO

Front End

Fulano

18

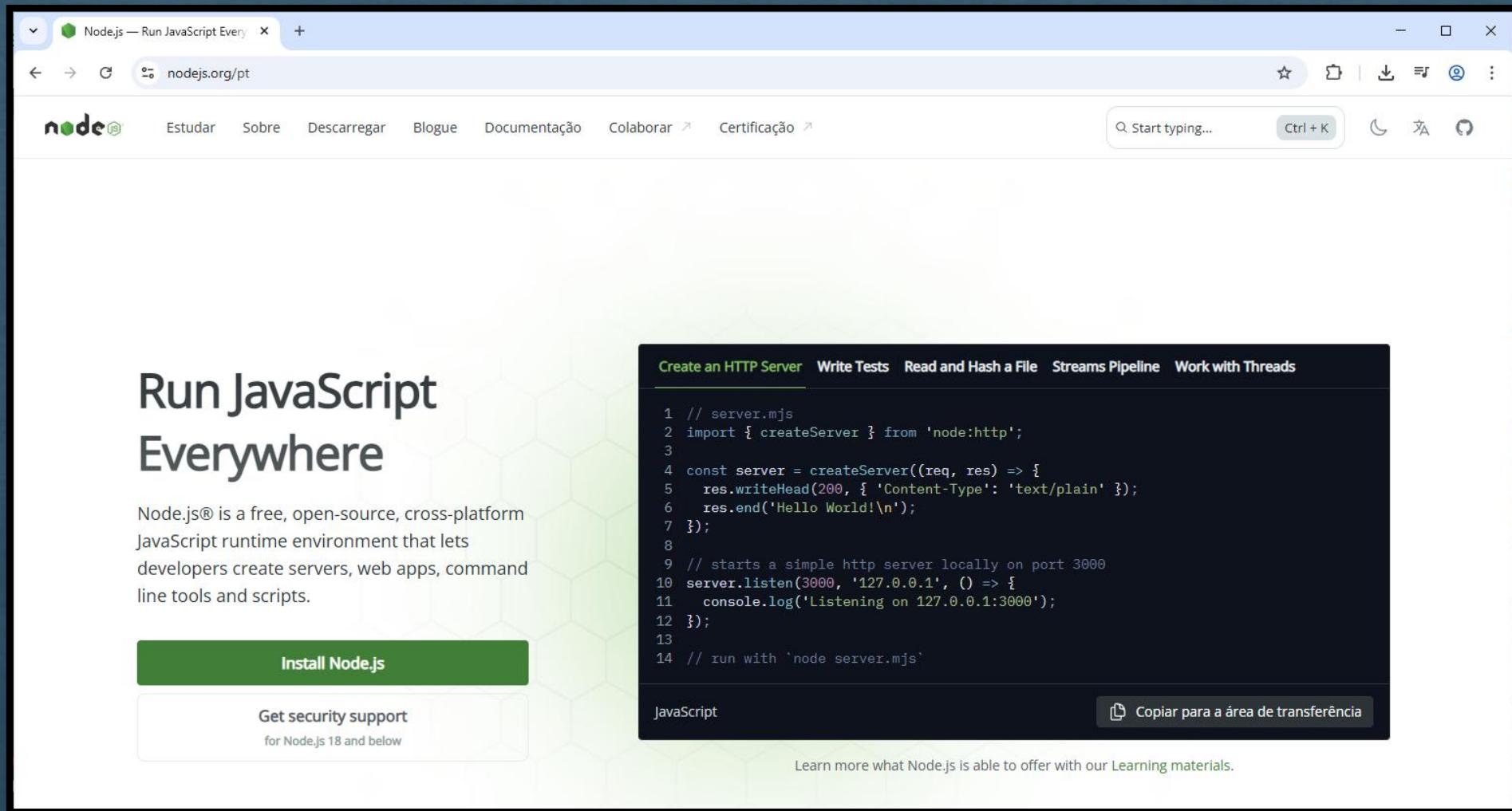
Cadastrar

João da Silva
37 anos

Maria Joaquina
45 anos

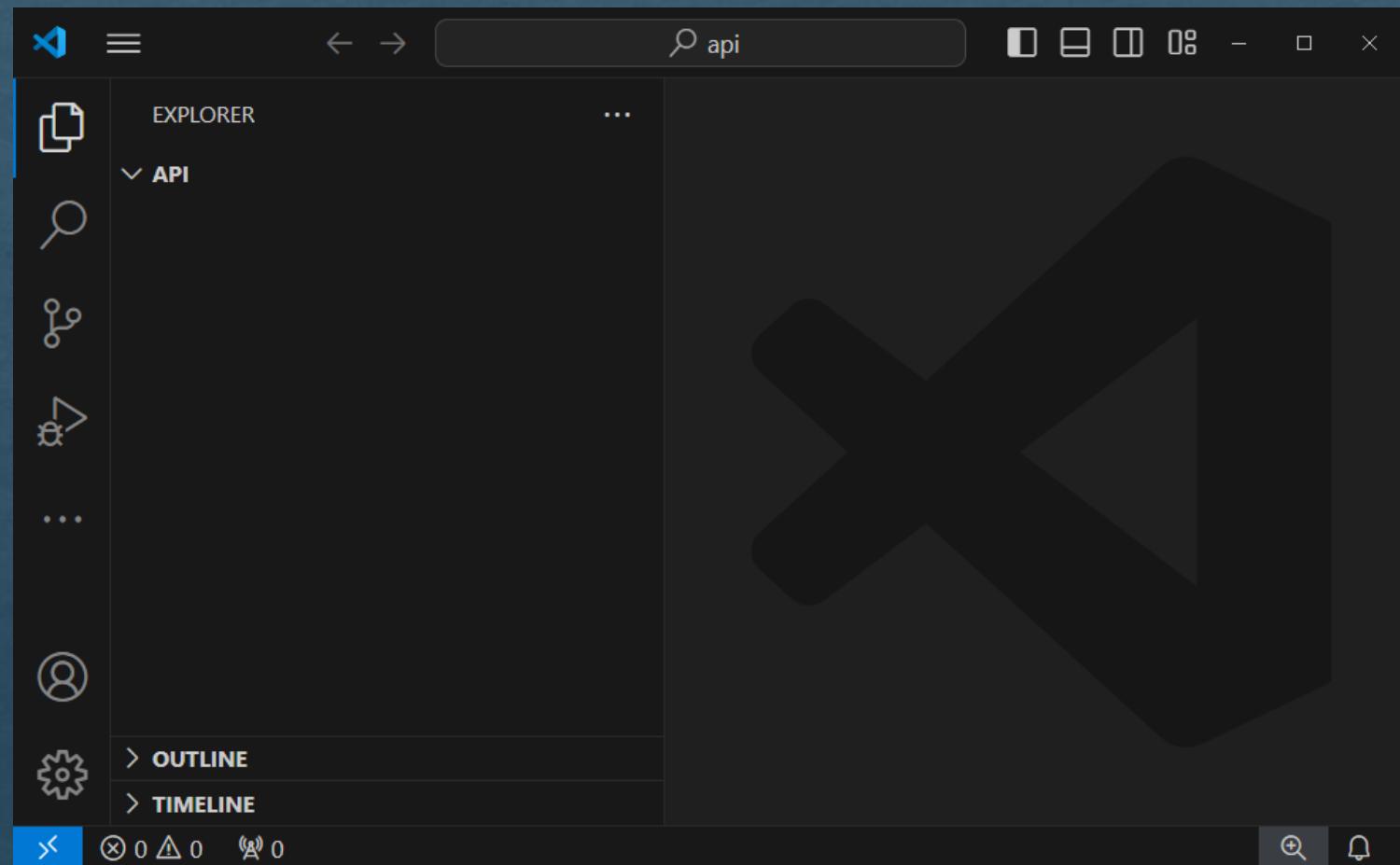
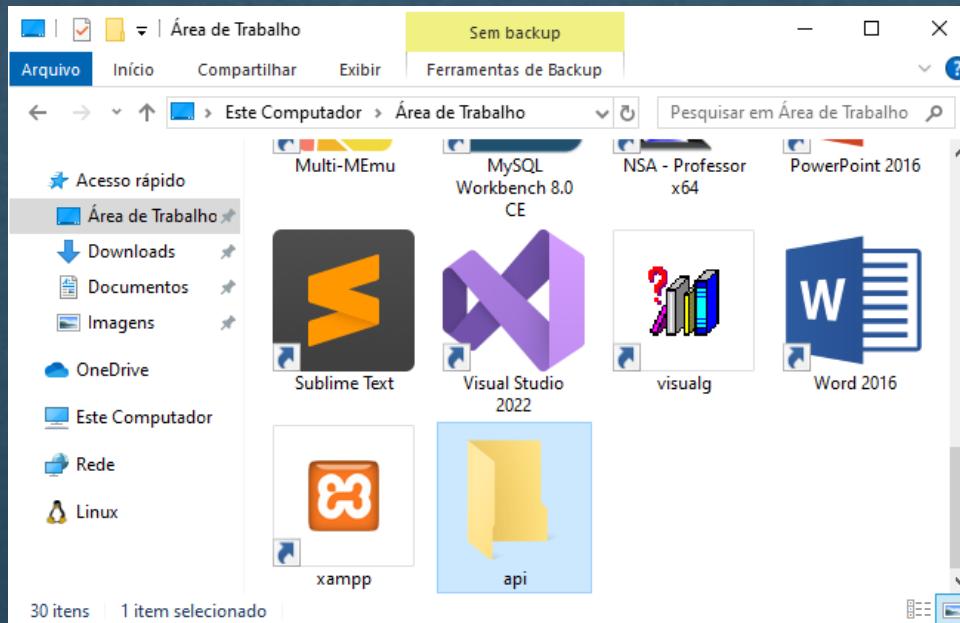
Mão na massa

- Para iniciarmos vamos precisar ter instalado na máquina o node.js (<https://nodejs.org/pt>)



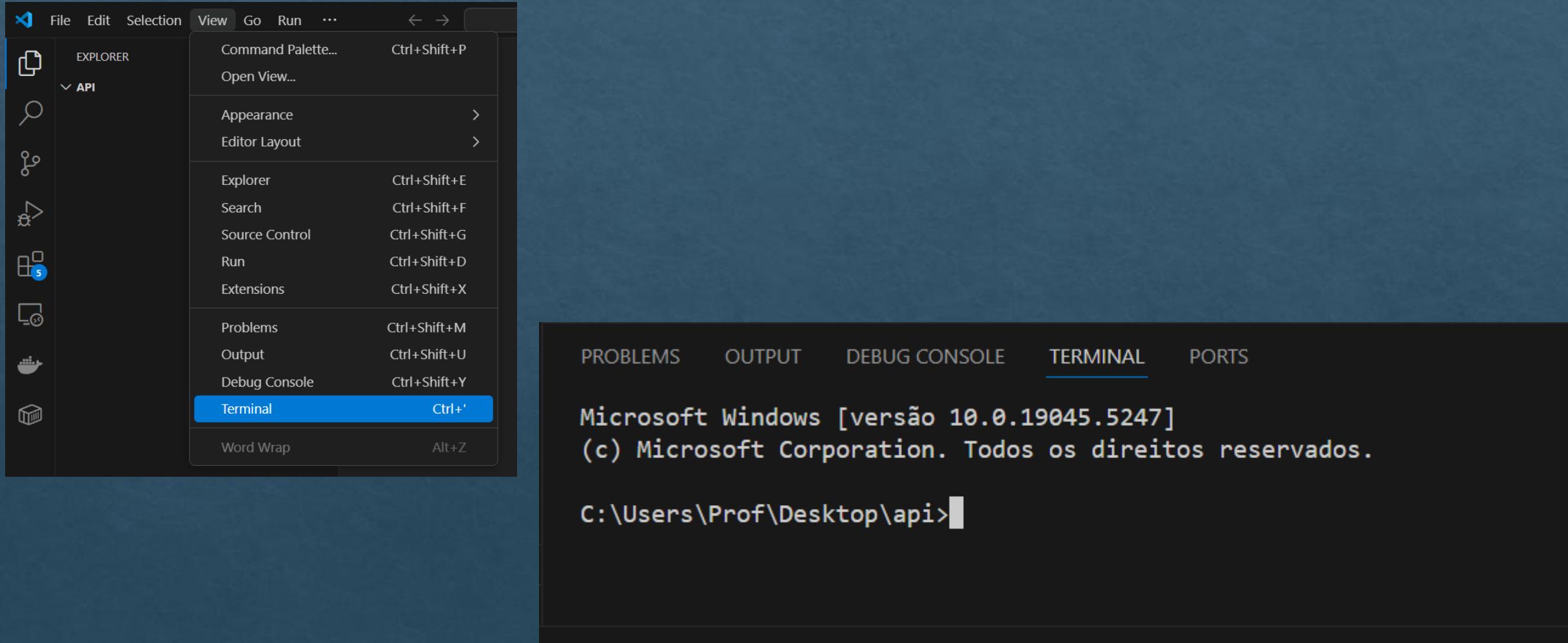
Mão na massa

- Crie uma pasta na área de trabalho chamada **api** e abra-a no VS Code



Mão na massa

- Abra uma janela de terminal no VS Code



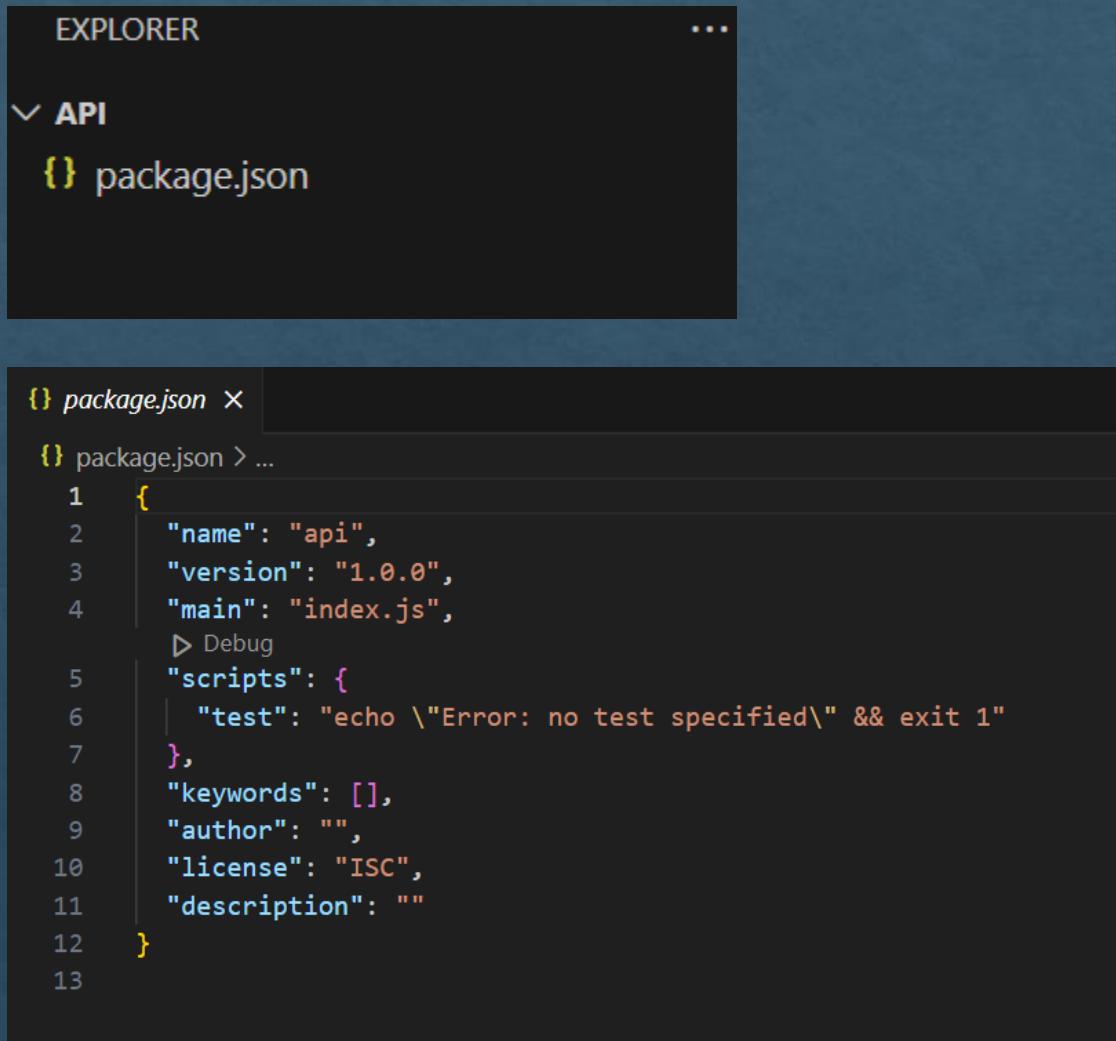
Mão na massa

- Inicie um projeto com o comando: `npm init -y`

```
C:\Users\Prof\Desktop\api>npm init -y
Wrote to C:\Users\Prof\Desktop\api\package.json:

{
  "name": "api",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

C:\Users\Prof\Desktop\api>
```

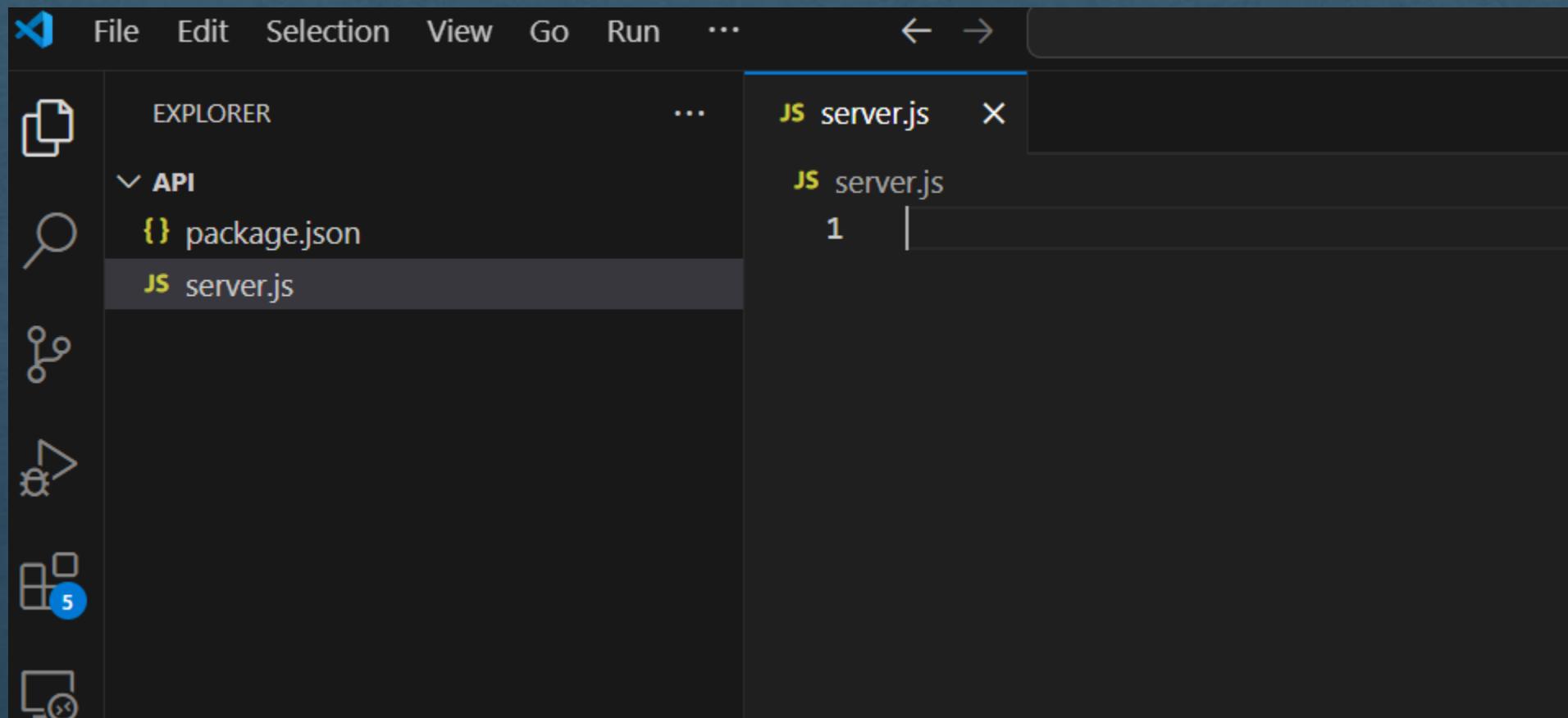


The image shows the VS Code interface. The Explorer pane on the left lists a single item: 'package.json'. The Editor pane on the right displays the contents of the package.json file, which is identical to the one shown in the terminal window above.

```
{
  "name": "api",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Mão na massa

- Crie um arquivo chamado: **server.js**



Mão na massa

- Instale a biblioteca: **express**
- Digite o comando: **npm i express**

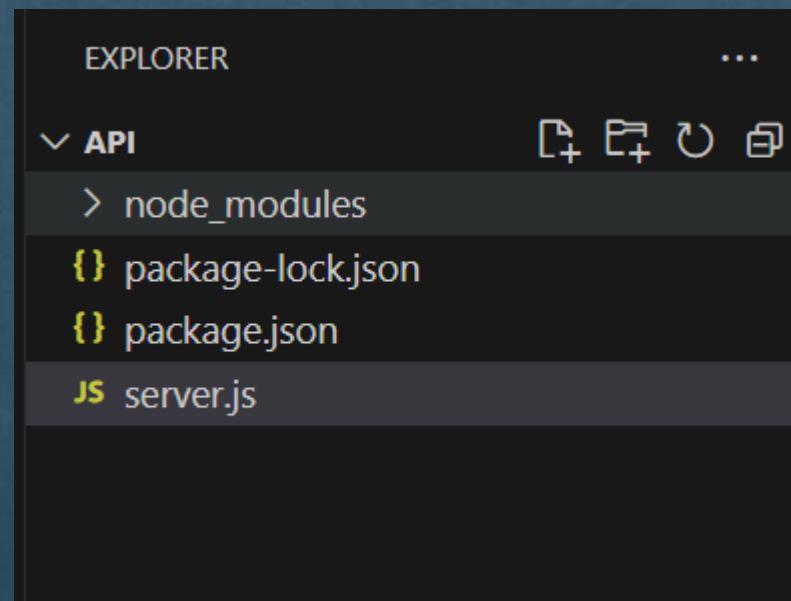
```
C:\Users\Prof\Desktop\api>npm i express

added 67 packages, and audited 68 packages in 3s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Prof\Desktop\api>
```



Http - Métodos

- GET → Listar
- POST → Criar
- PUT → Atualizar
- DELETE → Apagar

Http – Status de resposta

- **2xx → SUCESSO**
- **4xx → ERRO CLIENTE**
- **5xx → ERRO SERVIDOR**

Mão na massa

- Edite o arquivo: **server.js**

```
JS server.js > ...
1 import express from 'express'
2
3 const app = express()
4
5 //Criando uma rota
6 /*
7     1- Tipo de rota / Método http
8     2- Endereço
9 */
10 app.get('/usuarios', (req,res)=>{
11     res.send('Deu certo!')
12 })
13
14 //Informando a porta de resposta
15 app.listen(3000)
```

Mão na massa

- Para rodar o servidor digite: **node server.js**

```
C:\Users\Prof\Desktop\api>node server.js
(node:13860) Warning: To load an ES module, set "type": "module" in the package.json or use the .mjs extension.
(Use `node --trace-warnings ...` to show where the warning was created)
C:\Users\Prof\Desktop\api\server.js:1
import express from 'express'
^^^^^
```

Este erro ocorre, porque é necessário informar que o express irá ser usado como um módulo do node. Vamos alterar o arquivo **package.json**.

Mão na massa

- Altere o arquivo: **package.json**

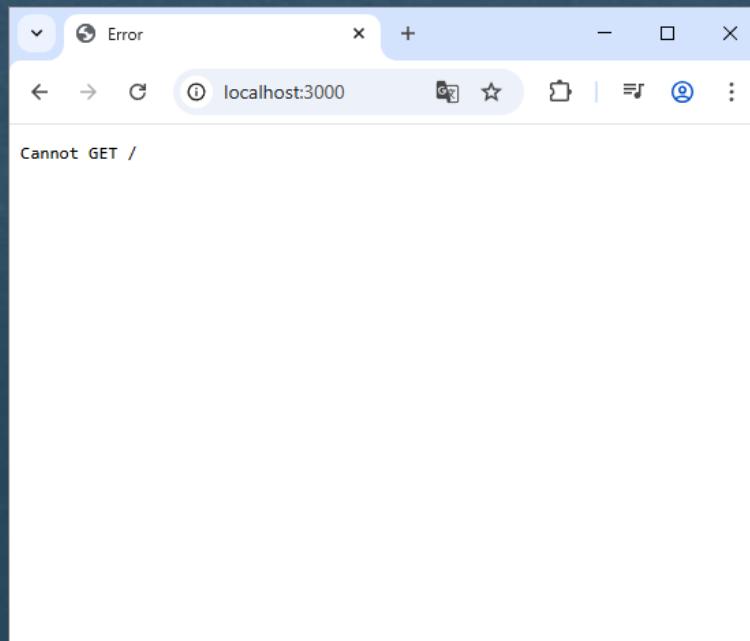
```
{ package.json > ...  
1  {  
2    "name": "api",  
3    "version": "1.0.0",  
4    "type": "module", ←  
5    "main": "index.js",  
6    "scripts": {  
7      "test": "echo \\\"Error: no test specified\\\" && exit 1"  
8    },  
9    "keywords": [],  
10   "author": "",  
11   "license": "ISC",  
12   "description": "",  
13   "dependencies": {  
14     "express": "^5.1.0"  
15   }  
16 }  
17
```

Mão na massa

- Rode novamente o servidor com o comando: **node server.js**

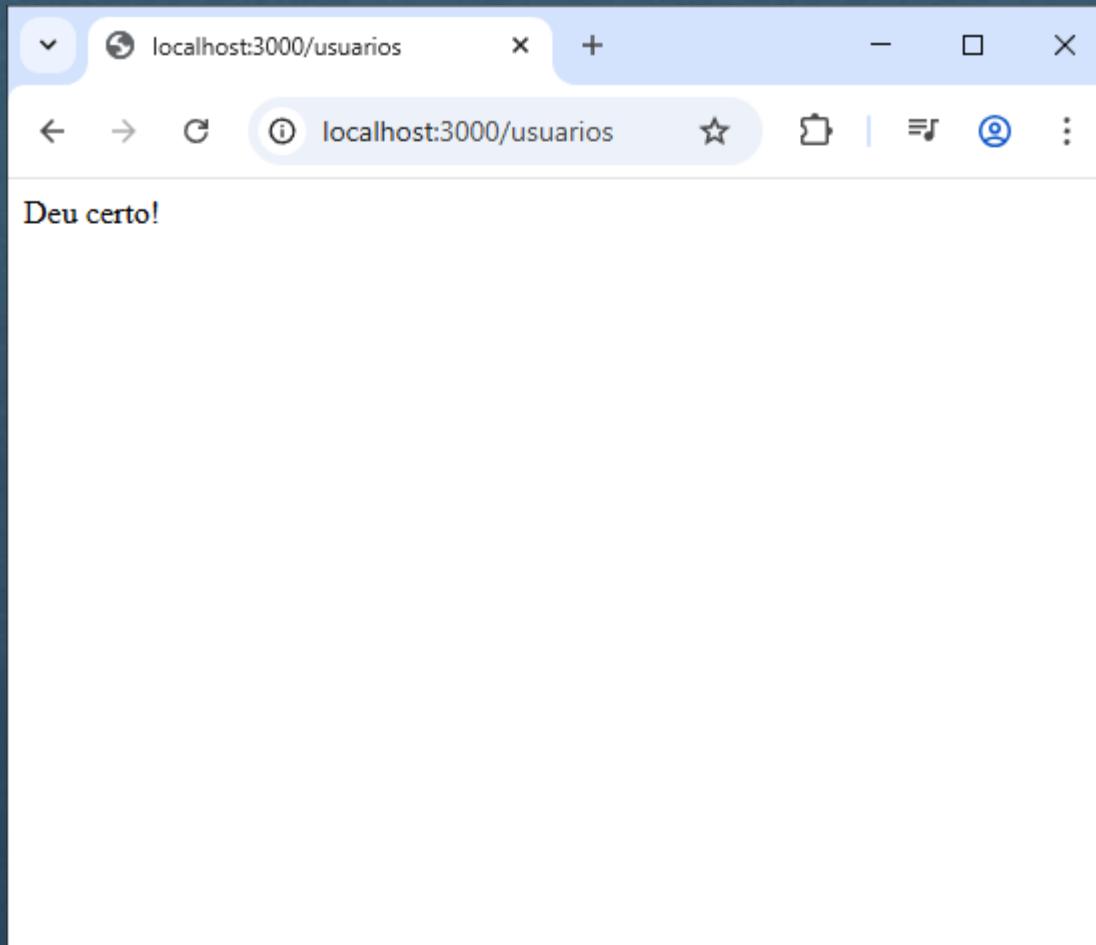
```
C:\Users\Prof\Desktop\api>node server.js
```

- Abra o navegador digitando na url: **localhost:3000**



Mão na massa

- Digite a rota correta: **localhost:3000/usuarios**



Mão na massa

- Vamos utilizar o Thunderclient para verificar as nossas rotas

The screenshot shows the Thunderclient application interface. On the left, there's a sidebar with tabs for 'Activity', 'Collections' (which is selected), and 'Env'. Below this is a search bar labeled 'filter collections' and a section for 'api com node' with a single entry: 'GET Acessando GET just now'. The main area has tabs for 'Query' (selected), 'Headers 2', 'Auth', 'Body', 'Tests', and 'Pre Run'. A 'Send' button is located at the top right of this section. To the right, the response details are shown: 'Status: 200 OK', 'Size: 10 Bytes', and 'Time: 3 ms'. The 'Response' tab is selected, displaying the message '1 Deu certo!'. At the bottom right are buttons for 'Response', 'Preview', 'Chart', and a download icon.

Mão na massa

- Próximos passos:
 - Criar um usuário
 - Listar usuários
 - Editar usuário
 - Deletar usuário

Mão na massa

- Adicionando uma variável para armazenar os usuários e criar a rota para criar um novo usuário

```
-  
3  const app = express()  
4  
5  const users = [] ←  
6  
7  //Criando uma rota  
8  /*  
9   | 1- Tipo de rota / Método http  
10  | 2- Endereço  
11 */  
12  
13  app.post('/usuarios', (req,res)=>{  
14    console.log(req)  
15    res.send('Ok POST') } ) ←  
16  
17  
18  app.get('/usuarios', (req,res)=>{  
19    res.send('Deu certo!') } )  
20  
21  
22  //Informando a porta de resposta  
23  app.listen(3000)
```

Mão na massa

- Crie uma nova requisição na mesma rota do tipo POST

The screenshot shows the Postman application interface. On the left, the request configuration is visible: a POST method, the URL `http://localhost:3000/usuarios`, and a blue "Send" button. Below the URL, there are tabs for "Query", "Headers 2", "Auth", "Body", "Tests", and "Pre Run". Under the "Query" tab, there is a table for "Query Parameters" with one entry: a checkbox labeled "parameter" and a text input labeled "value". On the right, the response details are shown: "Status: 404 Not Found" (which is circled in red), "Size: 148 Bytes", and "Time: 3 ms". Below these, there are tabs for "Response", "Headers 7", "Cookies", "Results", and "Docs". The "Response" tab is active, displaying the following HTML code:

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8">
5      <title>Error</title>
6    </head>
7    <body>
8      <pre>Cannot POST /usuarios</pre>
9    </body>
10   </html>
```

Este erro ocorreu, pois toda vez que é feita uma alteração no servidor, suas alterações precisam ser atualizadas, ou seja, reiniciar o servidor.
Pare o servidor e reinicie

Para que não seja necessário parar e reiniciar o servidor em todas as alterações de código, rode o servidor com o comando: **node –watch server.js**

REQUEST

- **Query Params (GET)** – Consultas
 - servidor.com/usuários?estado=sp&cidade=maua
- **Route Params (GET/PUT/DELETE)** – Buscar, deletar ou editar algo específico
 - get servidor/usuários/22
 - put servidor/usuários/22
 - delete servidor/usuários/22
- **Body Params (POST e PUT)**

{

 “**nome**”: “Anderson”,

 “**idade**” : 51

}

Mão na massa

- Altere a requisição para incluir no Body o corpo da requisição do tipo POST

The screenshot shows a POST request in the Postman application. The request URL is `http://localhost:3000/usuarios`. The 'Body' tab is selected, showing a JSON content type with the following data:

```
1  {
2      "nome": "Anderson",
3      "idade": 51,
4      "email": "anderson@email.com"
5  }
```

The response status is **200 OK**, size is **7 Bytes**, and time is **72 ms**. The response body shows a single item with the key **Ok** and value **POST**.

Mão na massa

- Visualize no terminal as informações que chegam, pesquise por body.

Detalhe:

Você não irá encontrar ainda estas informações, pois por padrão o express não utiliza o json. É necessário informar que iremos utilizar o arquivo no formato json.

Inclua no arquivo server.js uma linha:

```
JS server.js > ...
1  import express from 'express'
2
3  const app = express()
4
5  app.use(express.json()) ←
6
7  const users = []
8
9  //Criando uma rota
10 /*
11   |   1- Tipo de rota / Método http
12   |   2- Endereço
13 */
```

Mão na massa

- Utilizando o Thunderclient, faça novamente a requisição e pesquise por body

The screenshot shows the Thunderclient application interface. On the left, the request configuration is visible:

- Method: POST
- URL: http://localhost:3000/usuarios
- Body tab is selected.
- Content type: JSON
- JSON Content:

```
1  {
2    "nome": "Anderson",
3    "idade": 51,
4    "email": "anderson@email.com"
5 }
```

On the right, the response details are shown:

- Status: 200 OK
- Size: 7 Bytes
- Time: 95 ms
- Response tab is selected.
- Response body:

```
1 Ok POST
```

At the bottom, the terminal output shows the received body object:

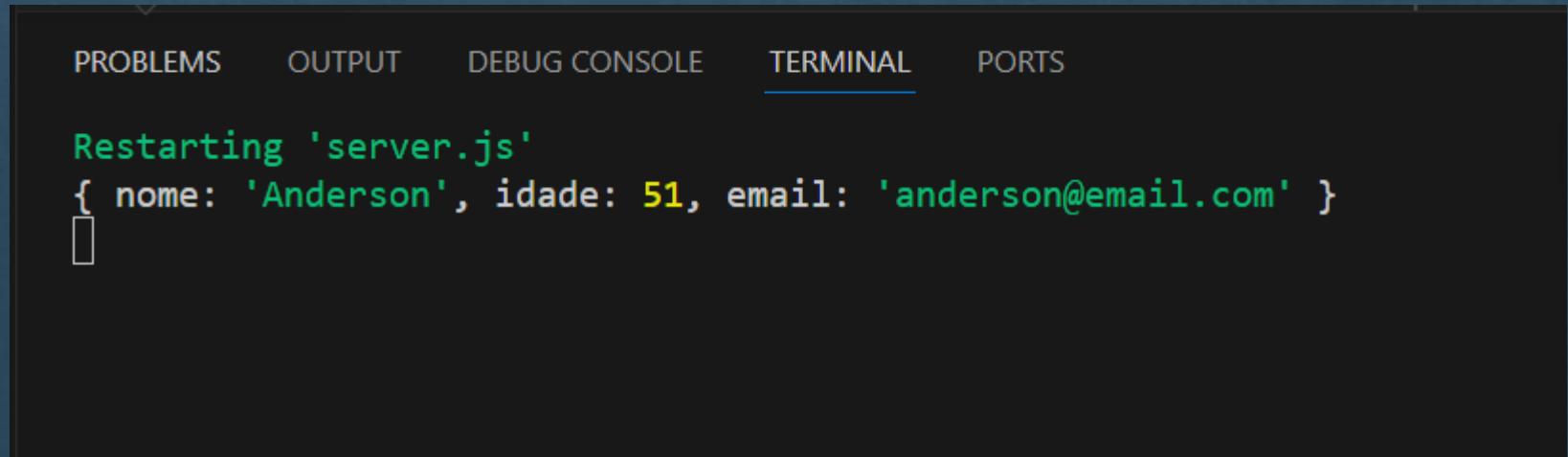
```
_raw: '/usuarios'
},
params: [Object: null prototype] {},
body: { nome: 'Anderson', idade: 51, email: 'anderson@email.com' },
length: undefined,
_eventsCount: 0,
route: Route {
  path: '/usuarios',
  stack: [ Layer ],
}
```

A red arrow points to the 'body' key in the terminal output, highlighting it.

Mão na massa

- Melhorando a saída, vamos colocar no console somente a parte que nos interessa: body.
- **Edito o arquivo server.js**

```
14  
15  app.post('/usuarios', (req,res)=>{  
16    |   console.log(req.body) ←  
17    |   res.send('Ok POST') ↗  
18  })  
19  
20  app.get('/usuarios', (req,res)=>{  
21    |   res.send('Deu certo!')
```



The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following text:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
  
Restarting 'server.js'  
{ nome: 'Anderson', idade: 51, email: 'anderson@email.com' }
```

Mão na massa

- Agora só precisamos adicionar à variável users (criada no código) as informações que estão chegando no body.
- **Altere o arquivo server.js**

```
14
15  app.post('/usuarios', (req,res)=>{
16    //console.log(req.body)
17    users.push(req.body) ←
18    res.send('Ok POST')
19  })
20
```

Mão na massa

- Já podemos, também, alterar a listagem dos usuários do método GET.
- Altere o arquivo server.js

```
15  app.post('/usuarios', (req,res)=>{
16    //console.log(req.body)
17    users.push(req.body)
18    res.send('Ok POST')
19  })
20
21  app.get('/usuarios', (req,res)=>{
22    res.json(users) ←
23  })
24
25  //Informando a porta de resposta
26  app.listen(3000)
```

Mão na massa

- Teste novamente, acessando a rota de post e depois a de get

The screenshot shows a POST request to `http://localhost:3000/usuarios`. The request body contains the following JSON data:

```
1 {  
2   "nome": "Anderson",  
3   "idade": 51,  
4   "email": "anderson@email.com"  
5 }
```

The response status is 200 OK, size is 7 Bytes, and time is 13 ms. The response body is:

```
1 Ok POST
```

The screenshot shows a GET request to `http://localhost:3000/usuarios`. The response status is 200 OK, size is 61 Bytes, and time is 3 ms. The response body is:

```
1 [  
2   {  
3     "nome": "Anderson",  
4     "idade": 51,  
5     "email": "anderson@email.com"  
6   }  
7 ]
```

Mão na massa

- Vamos incluir, também nas respostas, o status da resposta (2xx, 4xx, 5xx...).

```
15  app.post('/usuarios', (req,res)=>{  
16      //console.log(req.body)  
17      users.push(req.body)  
18      //res.send('Ok POST')  
19      res.status(201).json(req.body)  
20  })  
21  
22  app.get('/usuarios', (req,res)=>{  
23      res.status(200).json(users)  
24  })  
25  
26  //Informando a porta de resposta  
27  app.listen(3000)
```

The screenshot shows two separate Postman requests:

POST http://localhost:3000/usuarios

Body tab (selected): JSON Content

```
1 {  
2   "nome": "Maria",  
3   "idade": 34,  
4   "email": "maria@email.com"  
5 }
```

Response tab: Status: 201 Created, Size: 53 Bytes, Time: 21 ms

GET http://localhost:3000/usuarios

Query tab: Query Parameters

Response tab: Status: 200 OK, Size: 55 Bytes, Time: 2 ms

```
1 [  
2   {  
3     "nome": "Maria",  
4     "idade": 34,  
5     "email": "maria@email.com"  
6   }  
7 ]
```

Neste ponto já temos duas funcionalidades da nossa api
GET e POST, porém se o servidor for reiniciado, todas as
informações serão perdidas!

O nosso próximo passo será, agora, salvar estas
informações em um Banco de Dados.

Vamos utilizar o MongoDB.