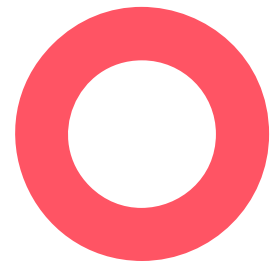


DESENVOLVIMENTO DE SISTEMAS

UC13

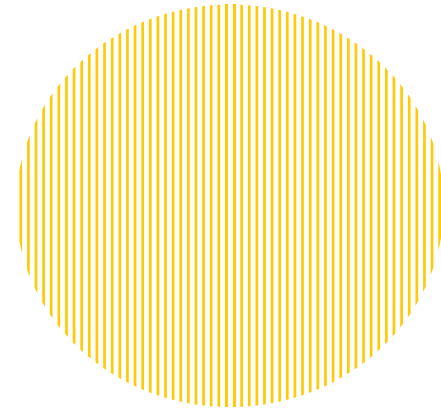
Prof. Viviane de Lima

viviane.lfrancelino@sp.senac.br



AULA 02


CONHECENDO O VALIDATOR




PROCURE PELA BIBLIOTECA VALIDATOR

Vamos usar a biblioteca validator, ela serve para validarmos se determinada string é e-mail, se é ip, se está tudo minúsculo ou maiúsculo dentre outras coisas. Observe que é possível ver a versão, quantidade de downloads, data de publicação e muito mais




 Nervous Penpal Message

ProductsPricingDocumentationCommunity







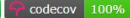


SearchSign UpSign In

validator 

13.7.0 • Public • Published 3 months ago

ReadmeExplore BETA0 Dependencies5.337 Dependents208 Versions

validator.js



A library of string validators and sanitizers.

Strings only

This library validates and sanitizes strings only.


If you're not sure if your input is a string, coerce it using `input + ''`. Passing anything other than a string will result in an error.

Installation and Usage


Install

```
> npm i validator
```

Repository


 github.com/validatorjs/validator.js

Homepage

 github.com/validatorjs/validator.js

Weekly Downloads

6.577.250



Version

License

MANUAL DE INSTALAÇÃO

This library validates and sanitizes strings only.

If you're not sure if your input is a string, coerce it using `input + ''`. Passing anything other than a string will result in an error.

Installation and Usage

Server-side usage

Install the library with `npm install validator`

No ES6

```
var validator = require('validator');

validator.isEmail('foo@bar.com'); // => true
```

ES6

```
import validator from 'validator';
```

Or, import only a subset of the library:

```
import isEmail from 'validator/lib/isEmail';
```

Tree-shakeable ES imports

```
import isEmail from 'validator/es/lib/isEmail';
```

Observe que temos um passo a passo de como realizar instalação de diversas formas

 github.com/validatorjs/validator.js

↓ Weekly Downloads

6.577.250



Version

13.7.0

License

MIT

Unpacked Size

646 kB

Total Files

209

Issues

121

Pull Requests

102

Last publish

3 months ago

Collaborators



➤ Try on RunKit

🚩 Report malware

INSTALAR O VALIDATOR

Com o seu projeto aberto, abra o terminal e digite **npm install validator**



12

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

```
PS C:\Users\Viviane\Downloads\node\projetos> npm install validator
```

```
added 1 package, and audited 3 packages in 972ms
```

```
found 0 vulnerabilities
```

```
PS C:\Users\Viviane\Downloads\node\projetos> 
```

INSTALAR O TYPES PARA O VALIDATOR

Vamos aproveitar e instalar o types para o validator, ou seja para ele autocompletar o comando de código quando estivermos usando esse validator. Digite o comando no terminal:

`npm install --save-dev @types/validator`

```
PS C:\Users\Viviane\Downloads\node\projetos> npm install --save-dev @types/validator
```

```
added 1 package, and audited 4 packages in 842ms
```

(colocamos `--save-dev` pois será usado apenas para o ambiente de desenvolvimento)

```
found 0 vulnerabilities
```

```
PS C:\Users\Viviane\Downloads\node\projetos> |
```

OBSERVE QUE NO PACKAGE.JSON TEMOS DEPENDÊNCIAS

EXPLORER

PROJETOS

- > node_modules
- src
 - TS index.ts
- { } package-lock.json
- { } package.json
- TS tsconfig.json

package.json > { } dependencies

```
1  {
2    "name": "projetos",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1",
8      "start-dev": "nodemon src/index.ts"
9    },
10   "author": "",
11   "license": "ISC",
12   "devDependencies": {
13     "@types/node": "^17.0.21"
14   },
15   "dependencies": {
16     "validator": "^13.7.0"
17   }
18 }
19
```

Veja que os types do validator também foram instalados

USANDO O VALIDATOR

EXPLORER

... {} package.json TS index.ts X

PROJETOS

> node_modules

> src

TS index.ts

{} package-lock.json

{} package.json

TS tsconfig.json

src > TS index.ts

```
1 //validto é um pacote do node_modules
2 import validator from 'validator'
3
4 //usando o validator
5 console.log(validator.isEmail('delimaviviane@yahoo.com.br'))
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS C:\Users\Viviane\Downloads\projetos> npm run start-dev

> projetos@1.0.0 start-dev

> nodemon src/index.ts

[nodemon] 2.0.15

[nodemon] to restart at any time, enter `rs`

[nodemon] watching path(s): *.*

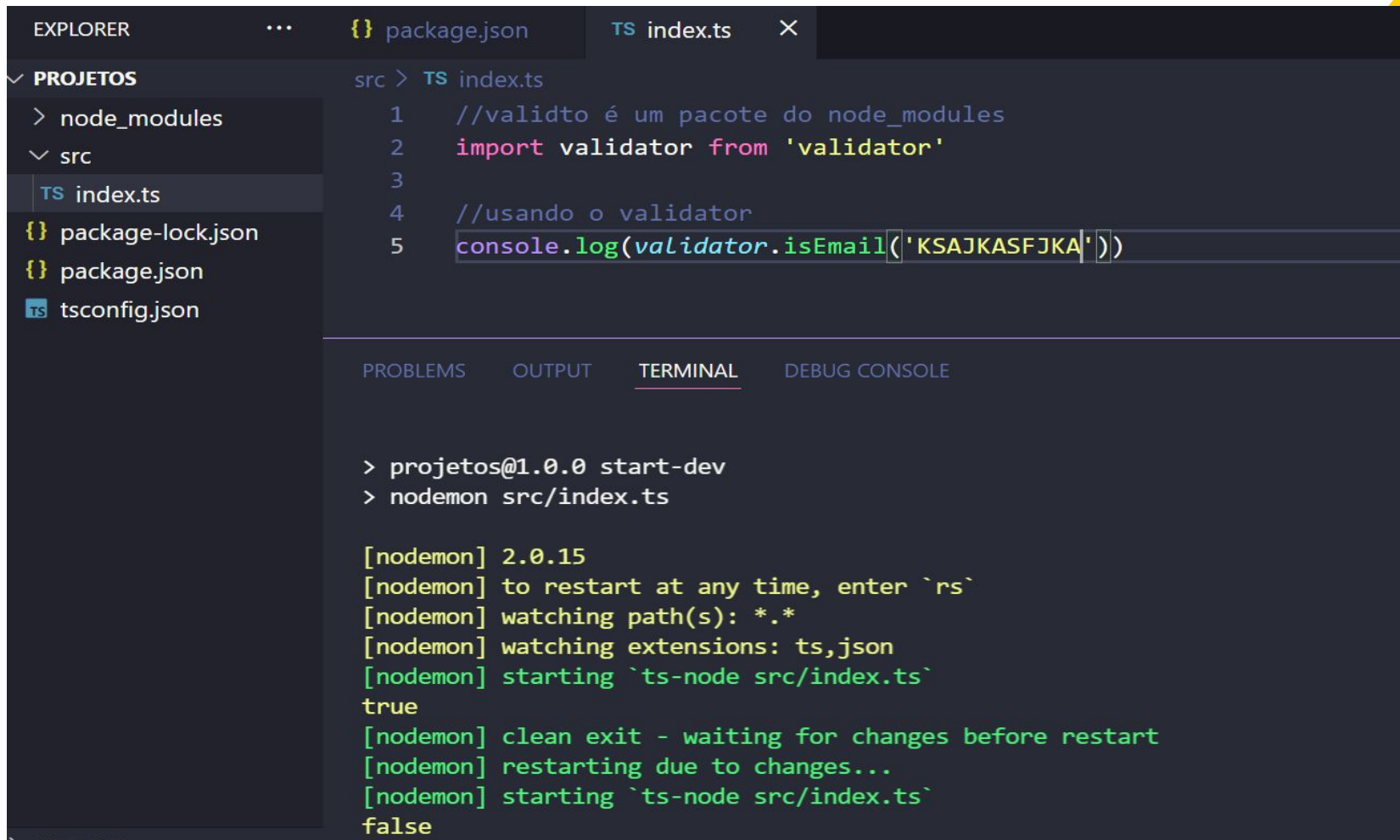
[nodemon] watching extensions: ts,json

[nodemon] starting `ts-node src/index.ts`

true

[nodemon] clean exit - waiting for changes before restart

VEJA QUE O QUE NÃO FOR PADRÃO E-MAIL ELE INFORMA QUE É



```
EXPLORER  ...  {} package.json  TS index.ts  X
```

✓ PROJETOS

- > node_modules
- ▼ src
 - TS index.ts
 - {} package-lock.json
 - {} package.json
 - tsconfig.json

```
src > TS index.ts
1  //validto é um pacote do node_modules
2  import validator from 'validator'
3
4  //usando o validator
5  console.log(validator.isEmail('KSAJKASFJKA'))
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
> projetos@1.0.0 start-dev
> nodemon src/index.ts

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: ts,json
[nodemon] starting `ts-node src/index.ts`
true
[nodemon] clean exit - waiting for changes before restart
[nodemon] restarting due to changes...
[nodemon] starting `ts-node src/index.ts`
false
```

VALIDAR UM IP

src > TS index.ts > ...

```
1  import validator from 'validator'
2
3  let ip = '127.0.0.1'
4
5  console.log(validator.isIP(ip))
6
```

PROBLEMS

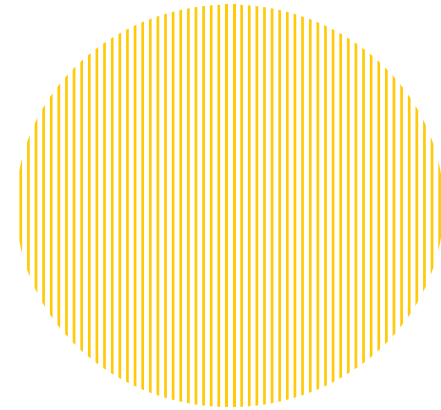
OUTPUT

TERMINAL

DEBUG CONSOLE

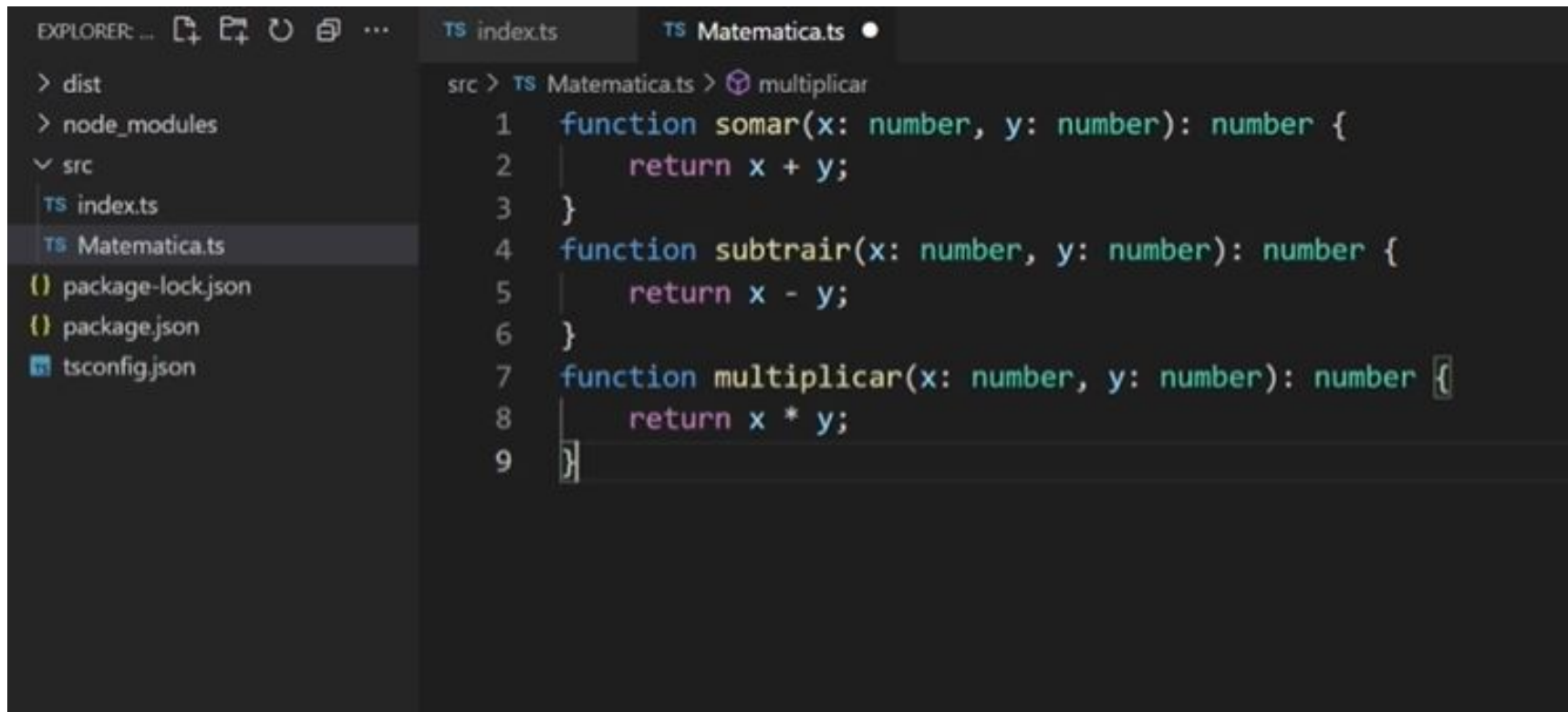
```
PS C:\Users\Viviane\Downloads\node\projetos> node dist/index.js
true
PS C:\Users\Viviane\Downloads\node\projetos> █
```

**IMPORT
E EXPORT**



IMPORTANDO UM ARQUIVO PRÓPRIO

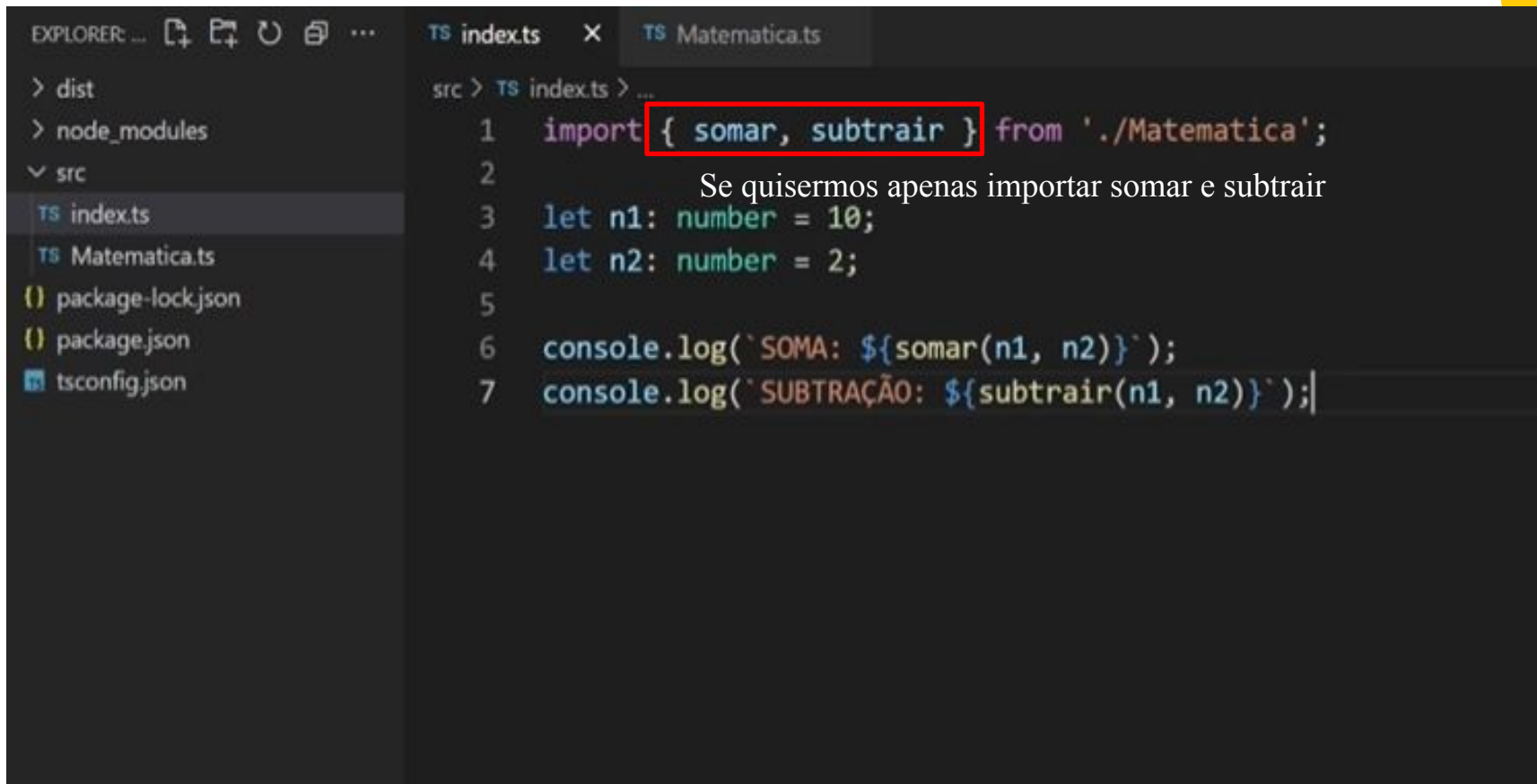
No npm além de conseguirmos instalar pacotes de terceiros(nodemon, validator etc) é possível importar nossos próprios arquivos, crie um arquivo chamado Matematica



```
EXPLORER: ... TS index.ts TS Matematica.ts
> dist
> node_modules
▼ src
  TS index.ts
  TS Matematica.ts
  package-lock.json
  package.json
  tsconfig.json

src > TS Matematica.ts > multiplicar
1  function somar(x: number, y: number): number {
2      |      return x + y;
3  }
4  function subtrair(x: number, y: number): number {
5      |      return x - y;
6  }
7  function multiplicar(x: number, y: number): number {
8      |      return x * y;
9  }
```

IMPORTE O ARQUIVO MATEMATICA.TS EM INDEX.TS



The image shows a VS Code editor window with two tabs: 'TS index.ts' and 'TS Matematica.ts'. The 'TS index.ts' tab is active, displaying the following code:

```
1 import { somar, subtrair } from './Matematica';  
2  
3 let n1: number = 10;  
4 let n2: number = 2;  
5  
6 console.log(`SOMA: ${somar(n1, n2)}`);  
7 console.log(`SUBTRAÇÃO: ${subtrair(n1, n2)}`);
```

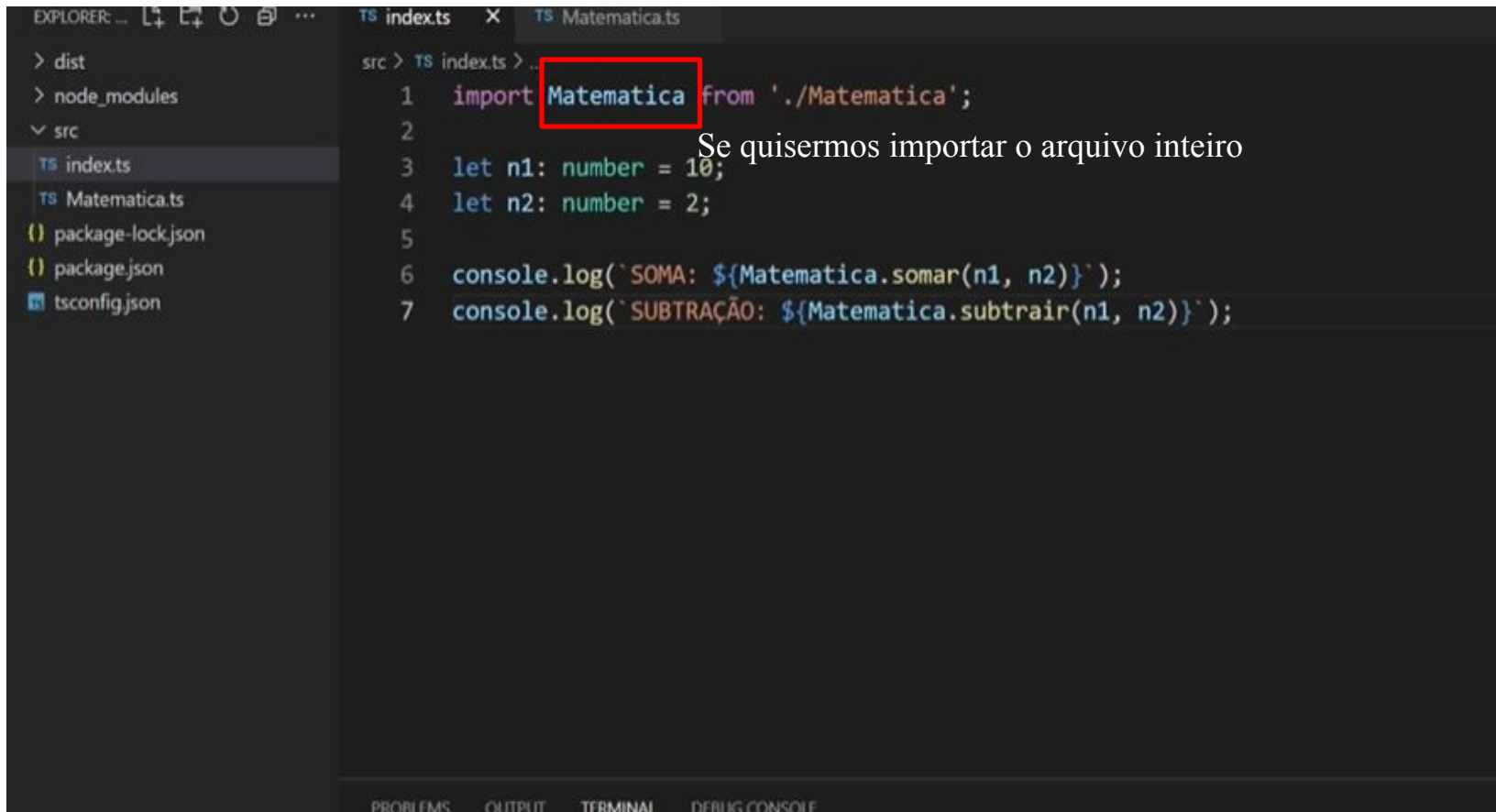
A red rectangle highlights the import statement on line 1: `import { somar, subtrair } from './Matematica';`. A text annotation is placed over the code on line 2:

Se quisermos apenas importar somar e subtrair

The Explorer sidebar on the left shows the file structure:

- > dist
- > node_modules
- ▼ src
 - TS index.ts
 - TS Matematica.ts
- () package-lock.json
- () package.json
- tsconfig.json

IMPORTE O ARQUIVO MATEMATICA.TS EM INDEX.TS



The screenshot shows the Visual Studio Code editor with two files open: `TS index.ts` and `TS Matematica.ts`. The Explorer sidebar on the left shows the project structure with `src` containing `index.ts` and `Matematica.ts`. The main editor displays the content of `index.ts`, where the word `Matematica` in the `import` statement on line 1 is highlighted with a red box. A text annotation 'Se quisermos importar o arquivo inteiro' points to this box. The code in `index.ts` is as follows:

```
1 import Matematica from './Matematica';
2
3 let n1: number = 10;
4 let n2: number = 2;
5
6 console.log(`SOMA: ${Matematica.somar(n1, n2)}`);
7 console.log(`SUBTRAÇÃO: ${Matematica.subtrair(n1, n2)}`);
```

The bottom of the editor shows tabs for PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE.

EXPORTANDO MATEMATICA.TS



Para que o arquivo matematica.ts funcione em index.ts precisamos importar o arquivo matematica.ts

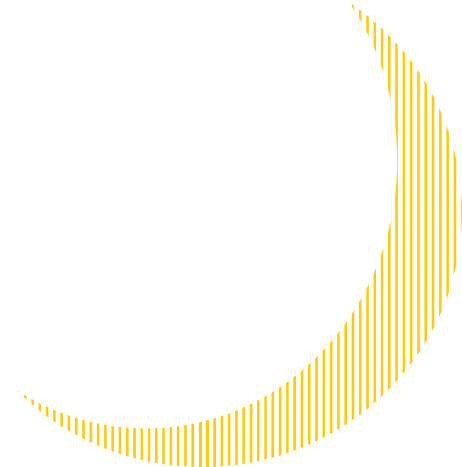
```
2
3 function somar(x: number, y: number): number {
4     return x + y;
5 }
6 function subtrair(x: number, y: number): number {
7     return x - y;
8 }
9 function multiplicar(x: number, y: number): number {
10    return x * y;
11 }
12
13 export default {
14     somar,
15     subtrair,
16     multiplicar,
17 };
18
```



VAMOS PRATICAR?



Crie um arquivo chamado `filmes.ts`, dentro desse arquivo crie uma lista de filmes, em seguida importe esse arquivo dentro de `index.ts`



RESPOSTA

TS index.ts



TS filmes.ts



src > TS filmes.ts > [🔍] default

```
1  let filmes: string[] = ['Demon Slayer', 'Samurai X', 'Naruto', 'One Piece'];  
2  
3  export default filmes
```

RESPOSTA

TS index.ts



TS filmes.ts

src > TS index.ts

```
1  import validator from 'validator'
2  import matematica from './matematica'
3  import filmes from './filmes'
4
5
6  console.log(filmes)
```



INSTALANDO O EXPRESS E CRIANDO UM SERVIDOR

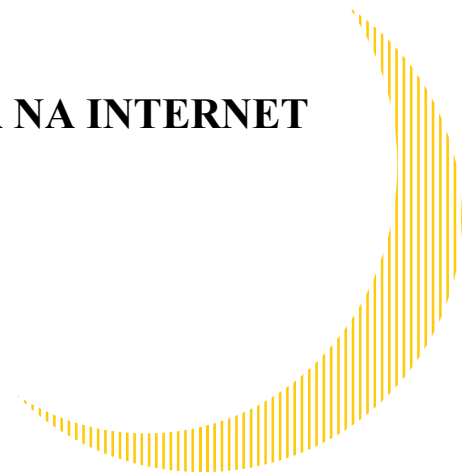


CRIE UM NOVO ARQUIVO CHAMADO SERVER.TS DENTRO DA PASTA SRC

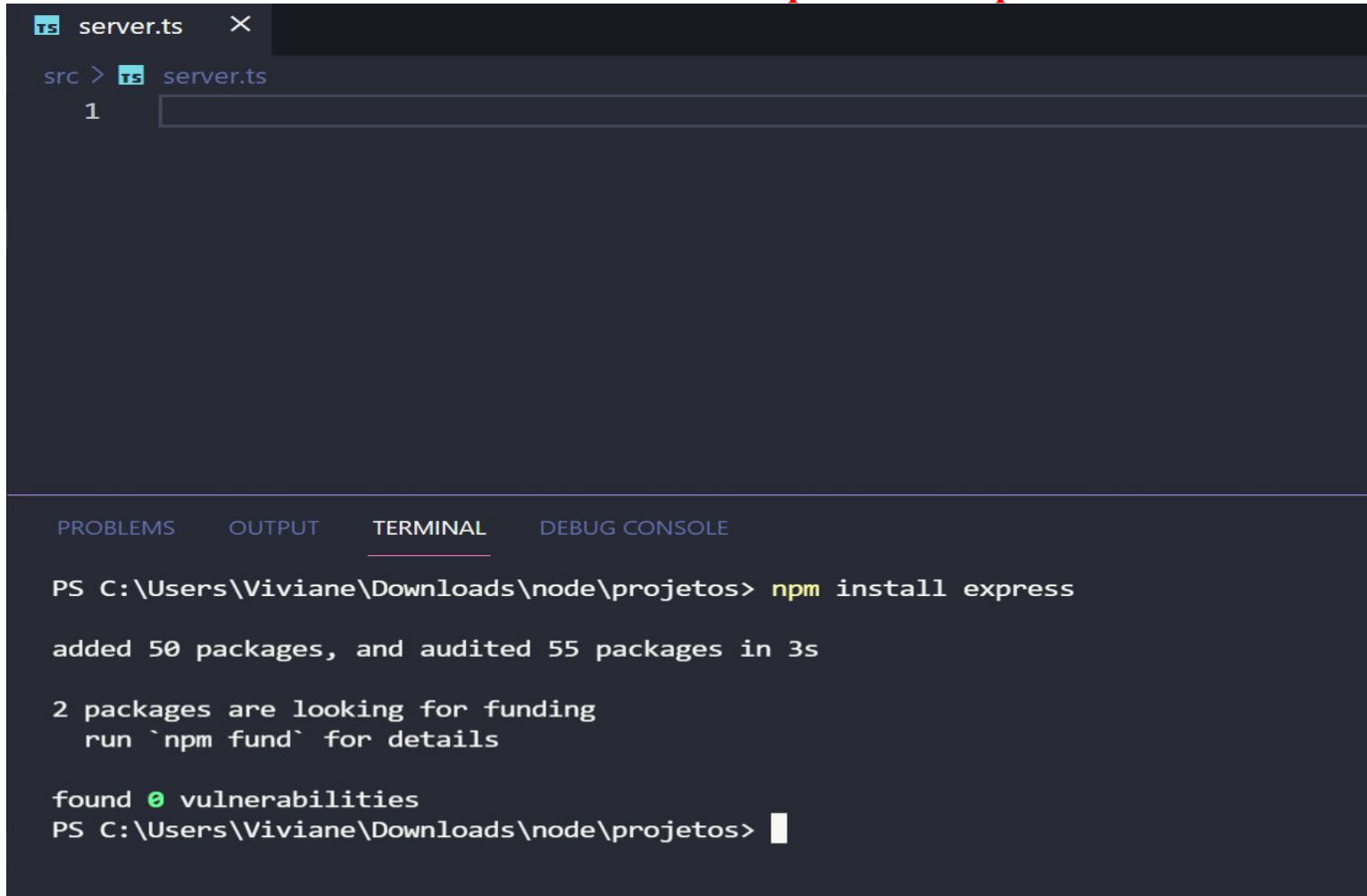


O arquivo server.ts será responsável por criar um servidor e deixar ele rodando em determinada porta do nosso computador, esse arquivo será responsável por toda a configuração do nosso servidor. Para conseguirmos fazer isso de forma tranquila iremos usar uma biblioteca chamada **Express**.

PESQUISE SOBRE O EXPRESS E VEJA O QUE VOCÊ ENCONTRA NA INTERNET



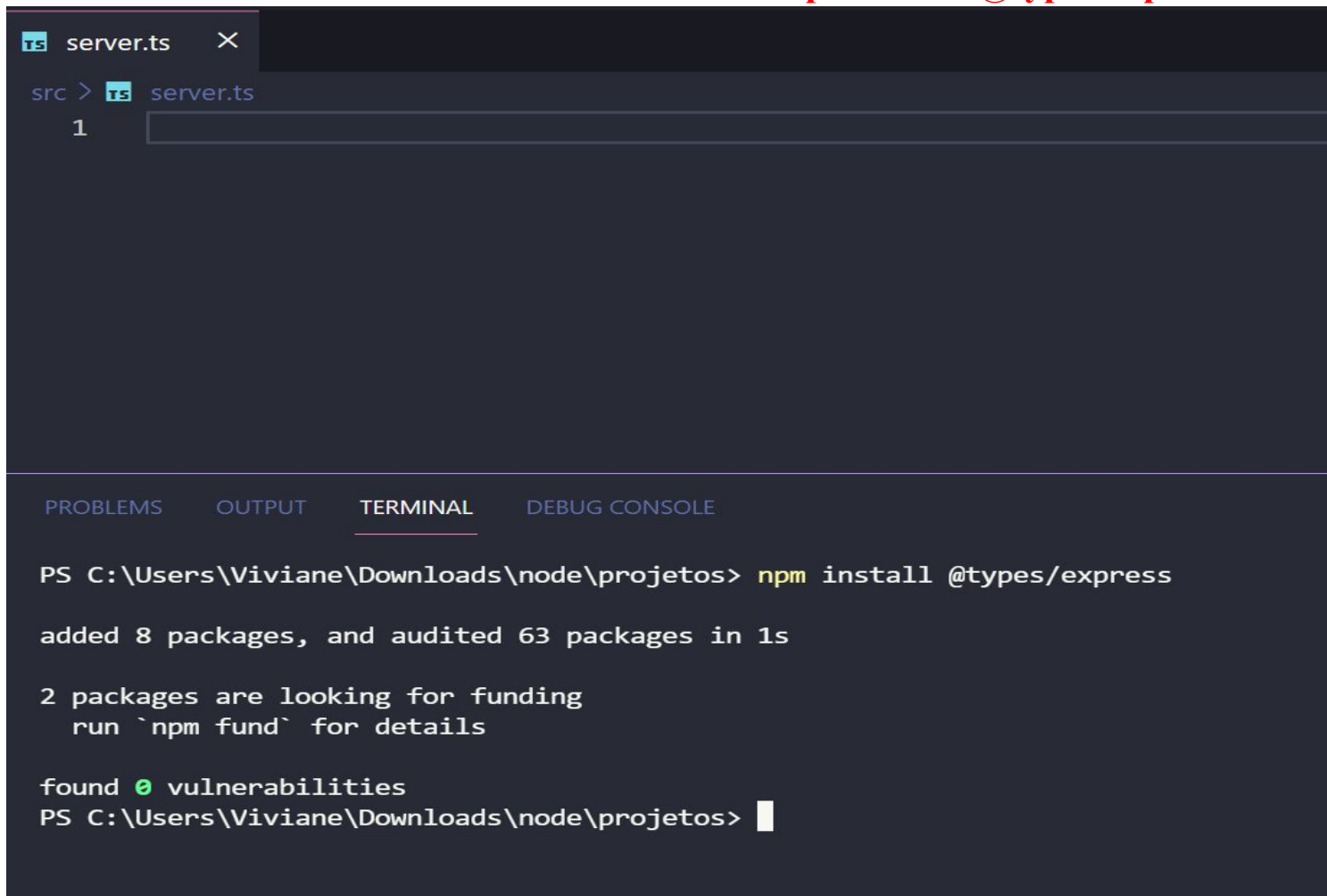
INSTALANDO O EXPRESS `npm install express`



The image shows a Visual Studio Code editor window with a file named `server.ts` open. The editor is in dark mode. Below the editor, the **TERMINAL** tab is active, displaying the output of the command `npm install express` run in a PowerShell prompt. The output shows that 50 packages were added and 55 packages were audited in 3 seconds. It also mentions that 2 packages are looking for funding and that no vulnerabilities were found.

```
src > server.ts  
1  
  
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  
  
PS C:\Users\Viviane\Downloads\node\projetos> npm install express  
  
added 50 packages, and audited 55 packages in 3s  
  
2 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities  
PS C:\Users\Viviane\Downloads\node\projetos> 
```

INSTALANDO O TYPES DO EXPRESS: `npm install @types/express`



The image shows a Visual Studio Code editor window with a dark theme. At the top, there is a tab labeled 'server.ts' with a TypeScript icon and a close button. Below the tab, the editor shows the file path 'src > server.ts' and a single line of code with the number '1' in the left margin. The bottom of the window features a panel with four tabs: 'PROBLEMS', 'OUTPUT', 'TERMINAL' (which is active and underlined), and 'DEBUG CONSOLE'. The terminal displays the output of the command 'npm install @types/express' executed in a PowerShell prompt. The output indicates that 8 packages were added and 63 packages were audited in 1 second. It also mentions that 2 packages are looking for funding and provides a command to run 'npm fund' for details. Finally, it states that 0 vulnerabilities were found. The terminal prompt is currently at 'PS C:\Users\Viviane\Downloads\node\projetos>'.

```
server.ts ×
src > server.ts
1

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

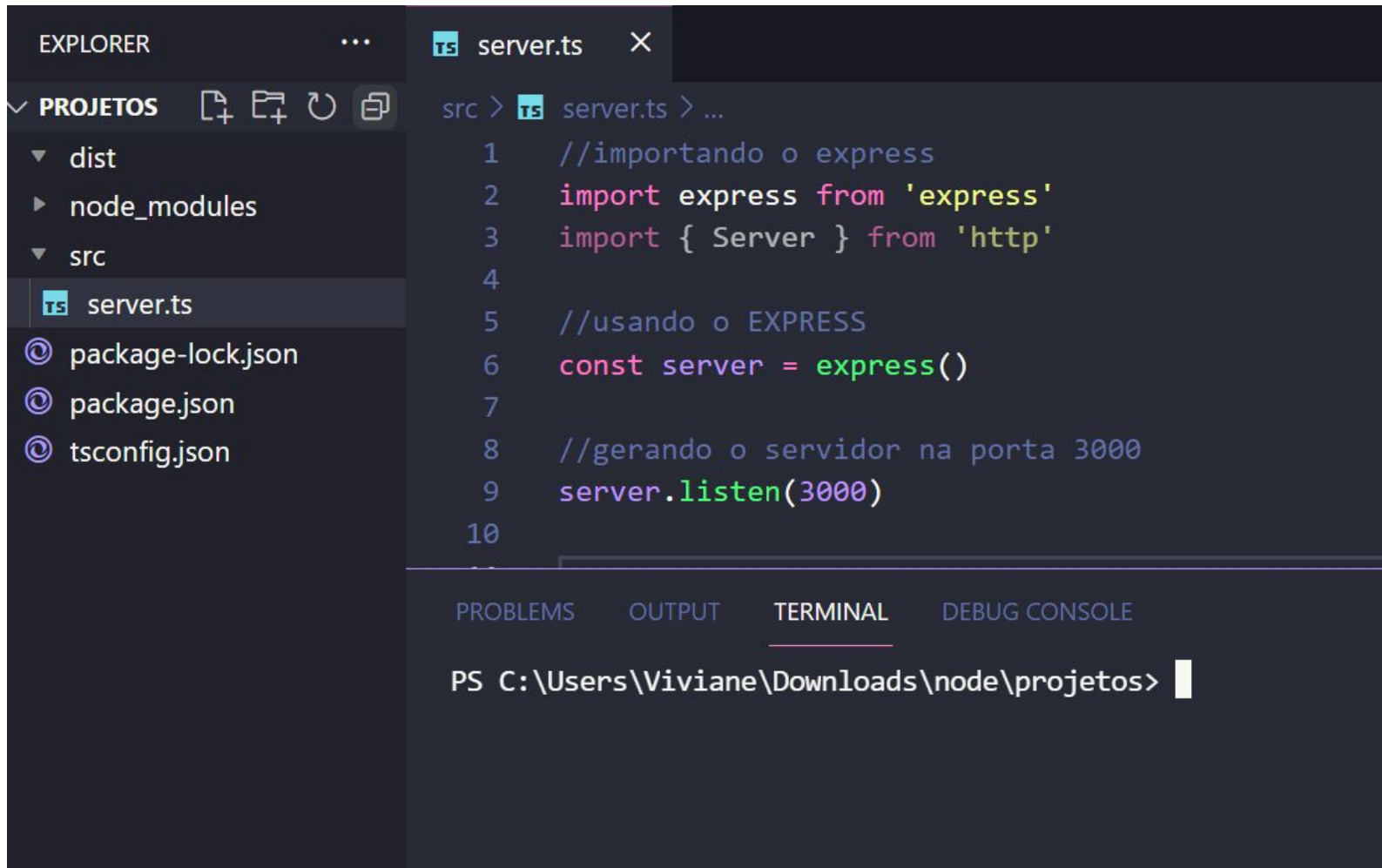
PS C:\Users\Viviane\Downloads\node\projetos> npm install @types/express

added 8 packages, and audited 63 packages in 1s

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

found 0 vulnerabilities
PS C:\Users\Viviane\Downloads\node\projetos> 
```

USANDO O EXPRESS



EXPLORER

PROJETOS

- dist
- node_modules
- src
 - server.ts
- package-lock.json
- package.json
- tsconfig.json

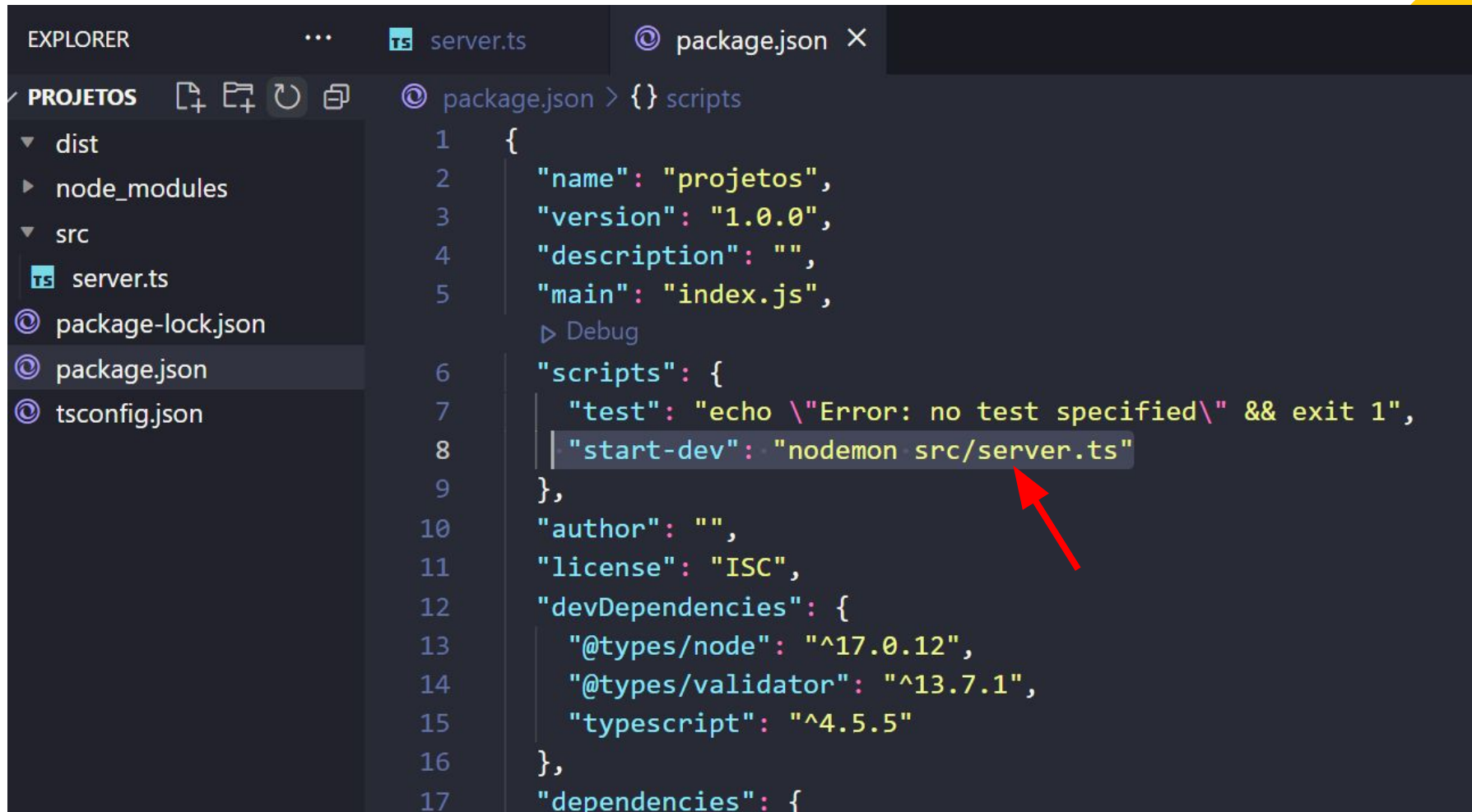
server.ts

```
src > server.ts > ...
1 //importando o express
2 import express from 'express'
3 import { Server } from 'http'
4
5 //usando o EXPRESS
6 const server = express()
7
8 //gerando o servidor na porta 3000
9 server.listen(3000)
10
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS C:\Users\Viviane\Downloads\node\projetos>

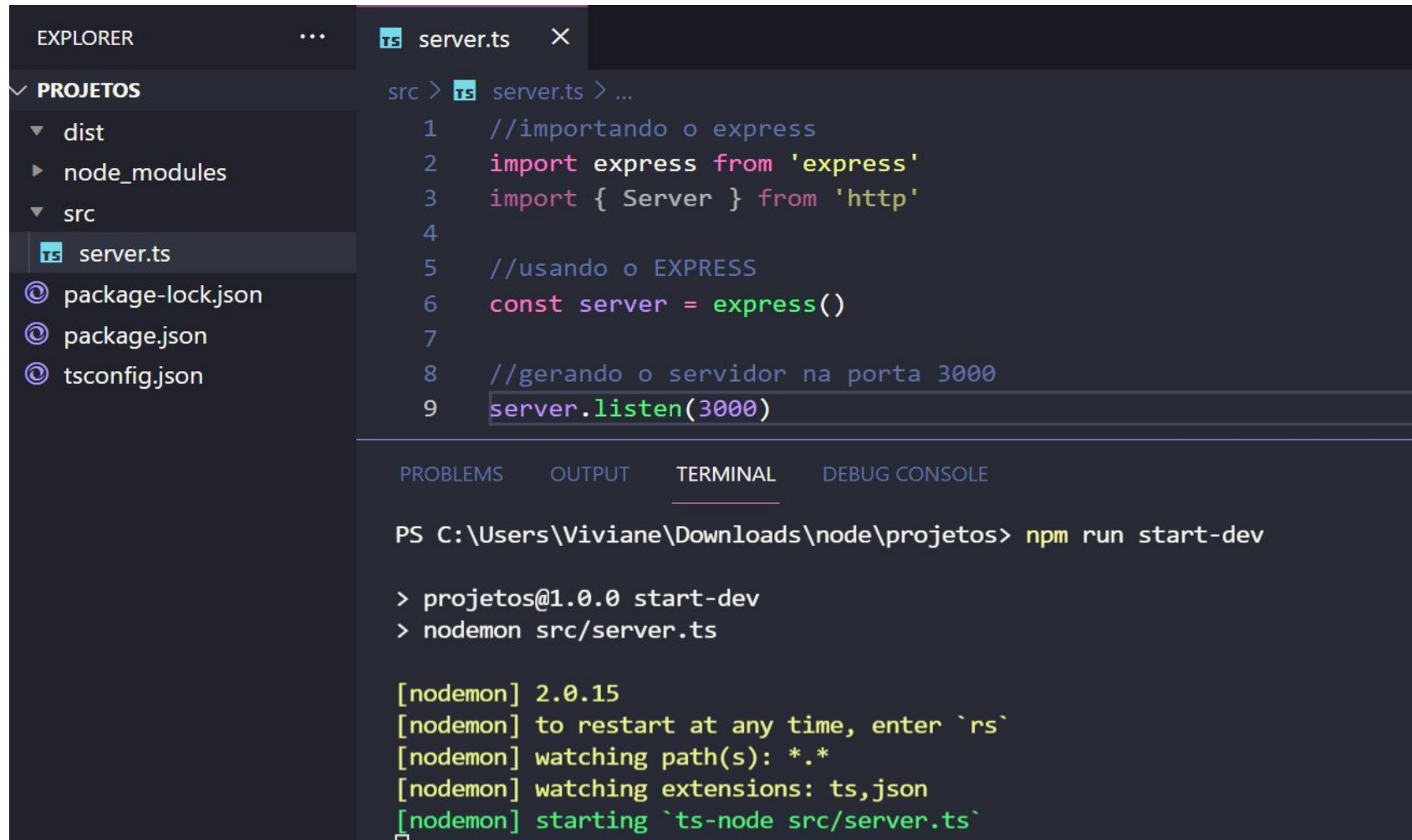
ALTERE O SCRIPT DO PACKAGE.JSON DE INDEX.TS PARA SERVER.TS



The image shows a Visual Studio Code editor window with the Explorer sidebar on the left and the package.json file open in the main editor. The Explorer sidebar shows the project structure with folders 'dist', 'node_modules', and 'src'. The 'src' folder is expanded, showing 'server.ts'. The main editor displays the 'package.json' file, which is a JSON object containing project metadata and scripts. The 'scripts' section is highlighted, and a red arrow points to the 'start-dev' script, which is set to 'nodemon src/server.ts'. The 'test' script is also visible, set to 'echo \"Error: no test specified\" && exit 1'. The 'devDependencies' section lists '@types/node', '@types/validator', and 'typescript'. The 'dependencies' section is partially visible at the bottom.

```
1  {
2    "name": "projetos",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1",
8      "start-dev": "nodemon src/server.ts"
9    },
10   "author": "",
11   "license": "ISC",
12   "devDependencies": {
13     "@types/node": "^17.0.12",
14     "@types/validator": "^13.7.1",
15     "typescript": "^4.5.5"
16   },
17   "dependencies": {
```


RODE O COMANDO NPM RUN START-DEV



The image shows a Visual Studio Code editor window. On the left, the Explorer sidebar shows a project named 'PROJETOS' with a 'src' folder containing 'server.ts'. The main editor area displays the contents of 'server.ts', which is a TypeScript file for an Express server. The code includes imports for 'express' and 'http', and a single line of code to start the server on port 3000. Below the editor, the TERMINAL tab is active, showing the command 'npm run start-dev' being executed. The terminal output shows that the command is successful, and the server is starting on port 3000.

```
EXPLORER  ...  TS server.ts X
```

PROJETOS

- dist
- node_modules
- src
 - TS server.ts
- package-lock.json
- package.json
- tsconfig.json

```
src > TS server.ts > ...
1  //importando o express
2  import express from 'express'
3  import { Server } from 'http'
4
5  //usando o EXPRESS
6  const server = express()
7
8  //gerando o servidor na porta 3000
9  server.listen(3000)
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS C:\Users\Viviane\Downloads\node\projetos> npm run start-dev

> projetos@1.0.0 start-dev
> nodemon src/server.ts

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: ts,json
[nodemon] starting `ts-node src/server.ts`
```

VERIFICANDO SE O SERVIDOR ESTÁ SENDO EXECUTADO



Se apareceu isso, então seu servidor está funcionando, entretanto
Ele está dando erro, pois ele não encontrou a página principal

Vá na barra de endereços do seu navegador e digite
<http://localhost:3000>

CONFIGURANDO A PÁGINA PRINCIPAL



The image shows a screenshot of the Visual Studio Code editor. On the left, the Explorer sidebar is open, showing a project structure with folders 'dist', 'node_modules', and 'src'. Inside 'src', the file 'server.ts' is selected. Below the Explorer, a list of files is shown: 'package-lock.json', 'package.json', and 'tsconfig.json'. The main editor area displays the code for 'server.ts'. The code imports 'express' and 'http', creates an Express server, and configures a GET route for the root path ('/'). The route handler is an anonymous function that takes 'req' and 'res' as arguments. The server is then listened on port 80.

```
EXPLORER  ...  TS server.ts X

PROJETOS

▼ dist
▶ node_modules
▼ src
  TS server.ts
  package-lock.json
  package.json
  tsconfig.json

src > TS server.ts > ...

1  import express from 'express'
2  import { Server } from 'http'
3
4  const server = express()
5
6  /*vamos configurar nossa página principal
7  vamos colocar como segundo parâmetro uma função anônima
8  que vai receber dois parâmetros, um parâmetro chamado req
9  e outro chamado res */
10 server.get('/',(req,res)=>{
11
12 })
13
14 //server.listen sempre é o ultimo comando a ser executado
15 server.listen(80)
```

COMO ESTAMOS USANDO TYPESCRIPT PRECISAMOS TIPAR O REQ E O RES

TS server.ts X

src > TS server.ts > ...

```
1 //tipando req e res
2 import express,{Request, Response } from 'express'
3
4
5 import { Server } from 'http'
6
7 const server = express()
8
9 /*aqui vamos dizer que req é do tipo Request
10 e res é do tipo Response */
11 server.get('/',(req,res)=>{
12
13 })
14
15 //server.listen sempre é o ultimo comando a ser executado
16 server.listen(80)
17
```

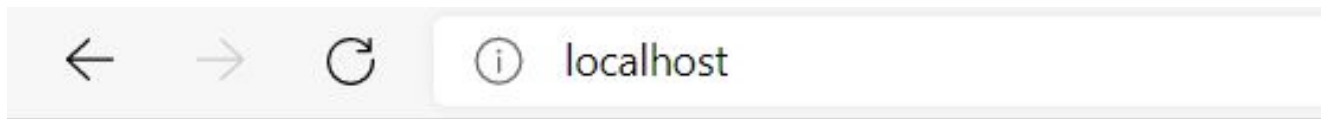
ENVIANDO UMA MENSAGEM AO SERVIDOR

server.ts

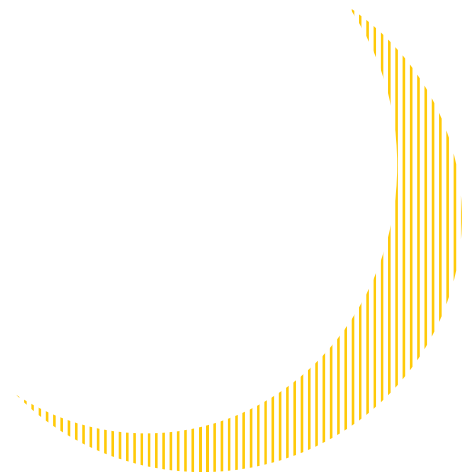
src > server.ts > ...

```
1
2  import express,{Request, Response } from 'express'
3
4
5  import { Server } from 'http'
6  import { send } from 'process'
7
8  const server = express()
9
10 //AQUI ESTOU USANDO APENAS O RES POR ENQUANTO
11 server.get('/',(req,res)=>{
12   |   res.send("Hello World!")
13 })
14
15
16 server.listen(80)
```

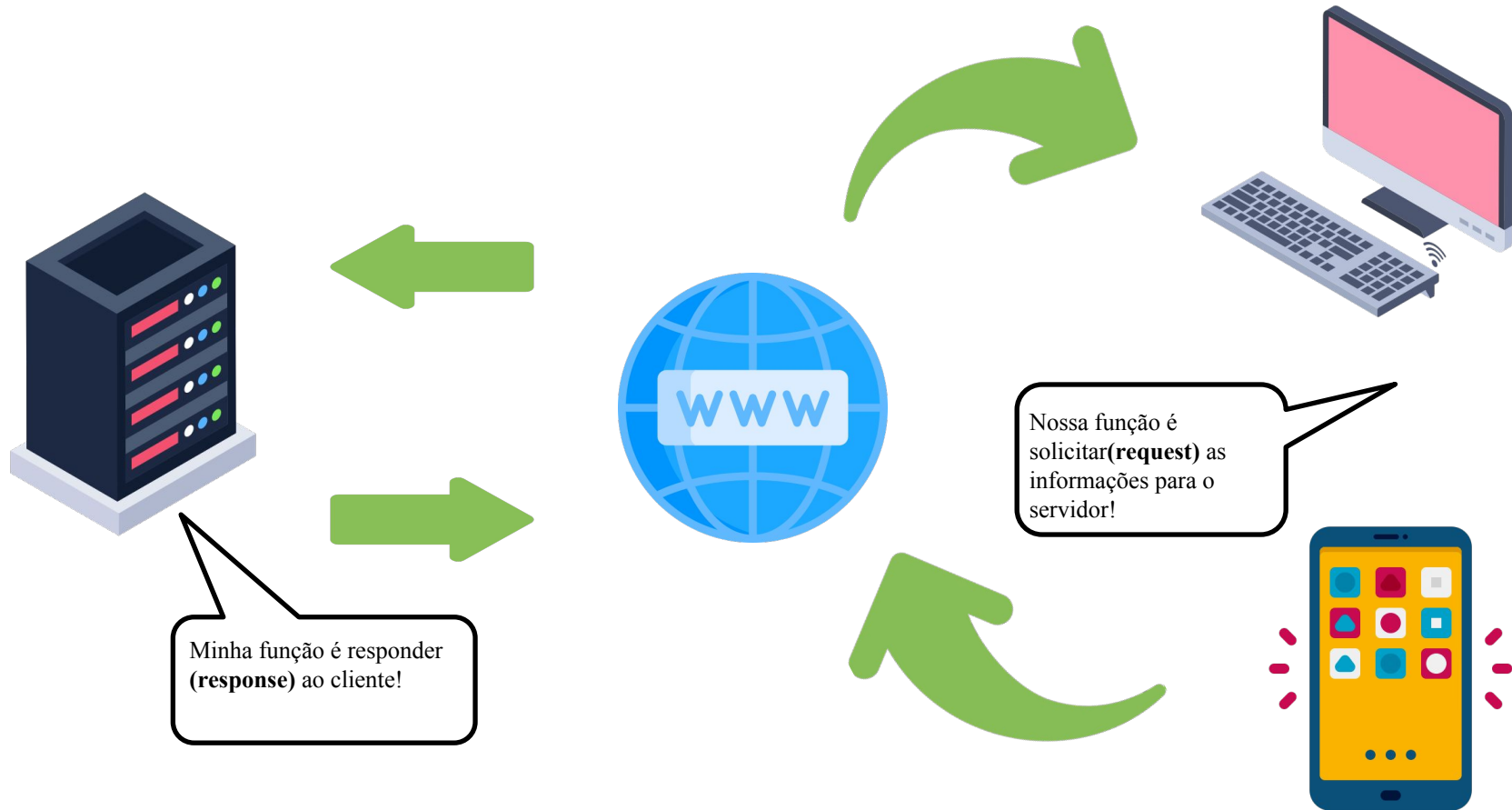
ATUALIZE SEU NAVEGADOR



Hello World!



REQUEST E RESPONSE



URL E URN

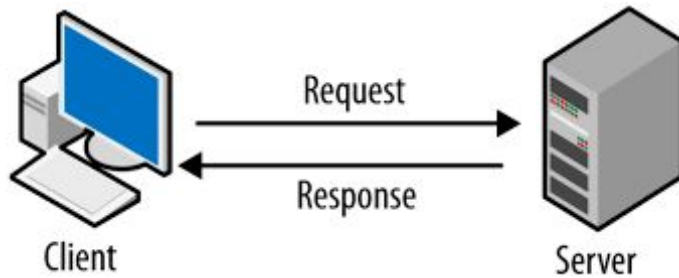
URL

Indica **onde** se encontra o recurso que desejamos obter e sempre começa com um **protocolo**. Neste caso, HTTP.

 **http://**www.digitalhouse.com/br/perguntas-frequentes

URN

É o **nome exato** do recurso uniforme.
O nome do domínio e, algumas vezes, o nome do recurso.



MÉTODOS HTTP

GET

PUT

POST

DELETE