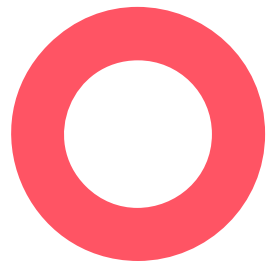


# DESENVOLVIMENTO DE SISTEMAS

## UC13

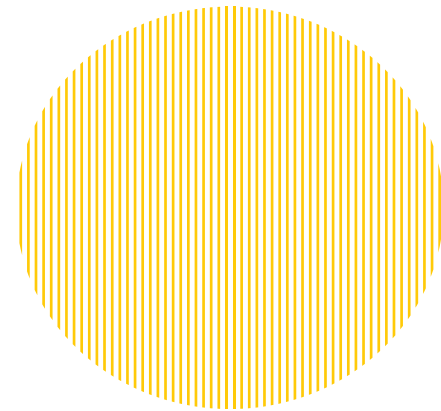
Prof. Viviane de Lima

[viviane.lfrancelino@sp.senac.br](mailto:viviane.lfrancelino@sp.senac.br)

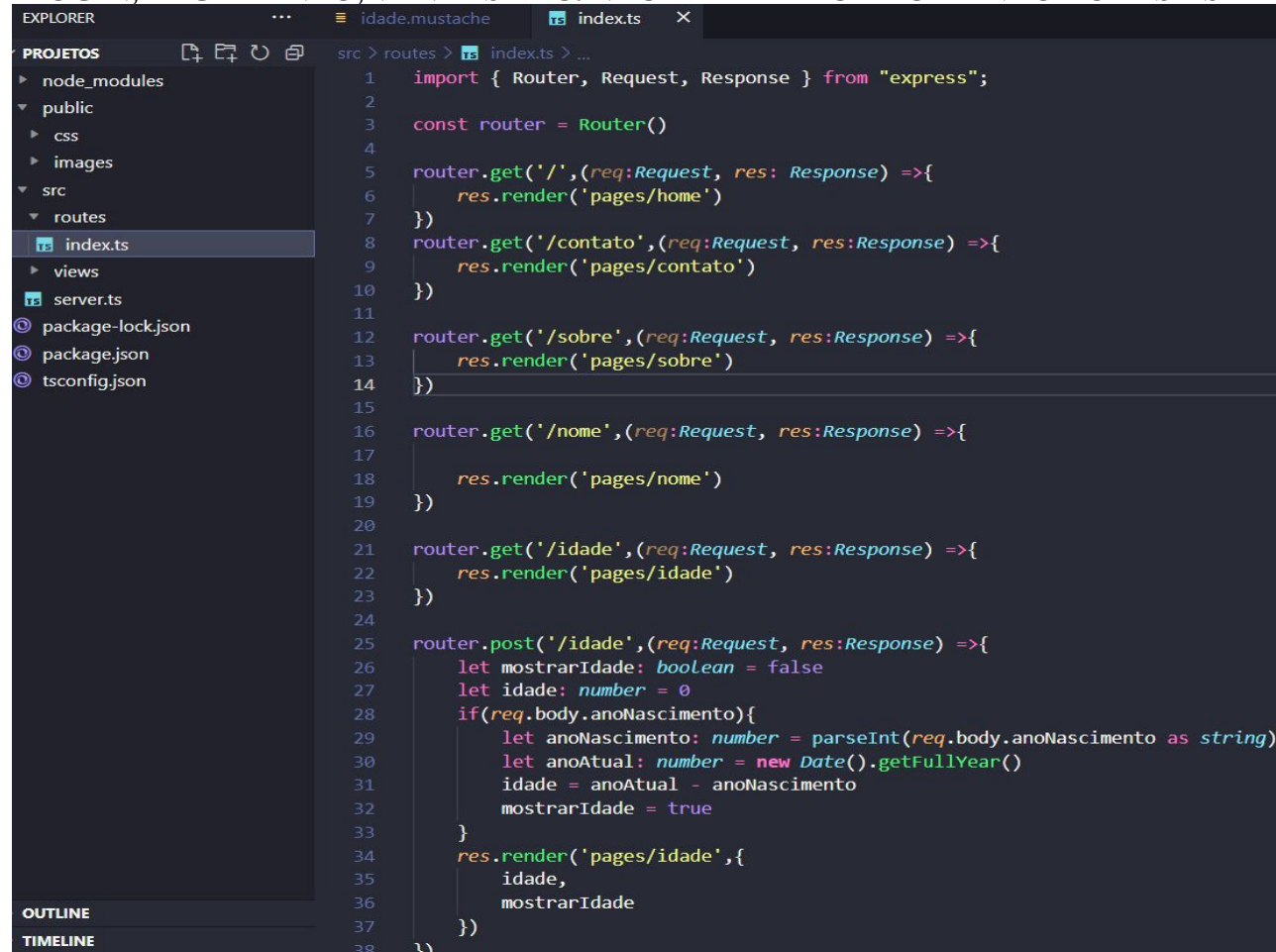


AULA 06

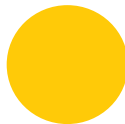
# INTRODUÇÃO À MVC



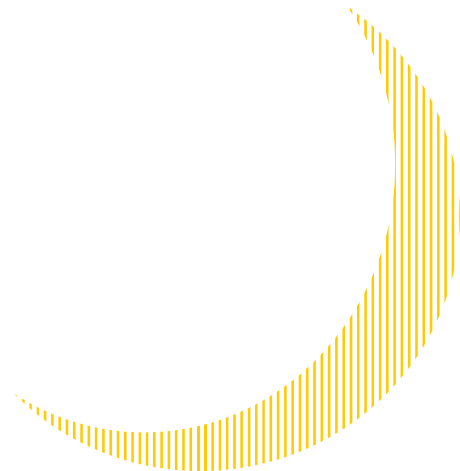
AGORA VAMOS AVANÇAR UM POQUINHO E MELHORAR A ORGANIZAÇÃO DOS NOSSO ARQUIVOS, POIS ESTAMOS FAZENDO TUDO EM INDEX.TS. IMAGINE QUE VOCÊ TEM UM SITE COMPLEXO COM LOGIN, PAGAMENTO, VENDAS ETC. NÃO DÁ PRA FICAR CRIANDO TODAS AS PÁGINAS EM UM



```
1  import { Router, Request, Response } from "express";
2
3  const router = Router()
4
5  router.get('/', (req: Request, res: Response) => {
6    res.render('pages/home')
7  })
8
9  router.get('/contato', (req: Request, res: Response) => {
10    res.render('pages/contato')
11  })
12
13  router.get('/sobre', (req: Request, res: Response) => {
14    res.render('pages/sobre')
15  })
16
17  router.get('/nome', (req: Request, res: Response) => {
18    res.render('pages/nome')
19  })
20
21  router.get('/idade', (req: Request, res: Response) => {
22    res.render('pages/idade')
23  })
24
25  router.post('/idade', (req: Request, res: Response) => {
26    let mostrarIdade: boolean = false
27    let idade: number = 0
28    if (req.body.anoNascimento) {
29      let anoNascimento: number = parseInt(req.body.anoNascimento as string)
30      let anoAtual: number = new Date().getFullYear()
31      idade = anoAtual - anoNascimento
32      mostrarIdade = true
33    }
34    res.render('pages/idade', {
35      idade,
36      mostrarIdade
37    })
38  })
```



**PORÉM NÃO É ERRADO FAZER DESSA FORMA, SE VOCÊ QUISER FAZER  
TUDO EM UM ARQUIVO SÓ, VÁ EM FRENTE. ENTRETANTO, A  
MANUTENÇÃO DO SEU CÓDIGO PODE SER PREJUDICADA, POIS VOCÊ  
LEVARÁ MAIS TEMPO PARA ENCONTRAR ROTAS,ERROS ETC**

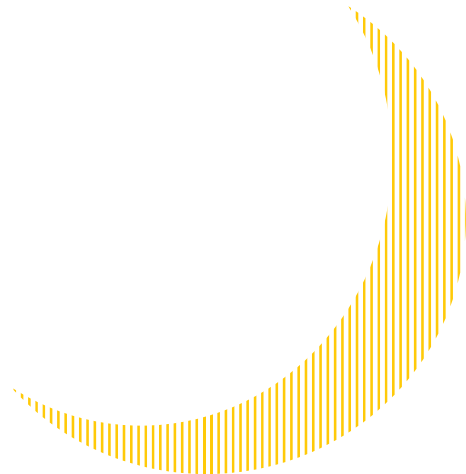


# O QUE É O M.V.C?

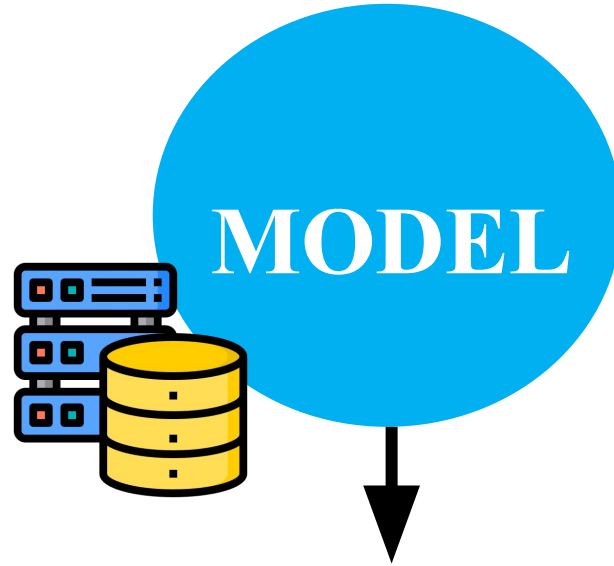


Quando desenvolvemos um sistema, ele possui várias classes internas e vários componentes, que em conjunto, desempenham várias funções. Por exemplo: exibição de dados do usuário, conexão com banco de dados, tratamento dos dados, modelagem, validações etc.

O **MVC** entra nesse contexto para propor divisão desses elementos, dividindo cada responsabilidade das aplicações dentro de suas camadas. **O MVC é um conceito, não é uma dependência**, ou seja, não vamos baixar o MVC, vamos implementar esse conceito no nosso código.

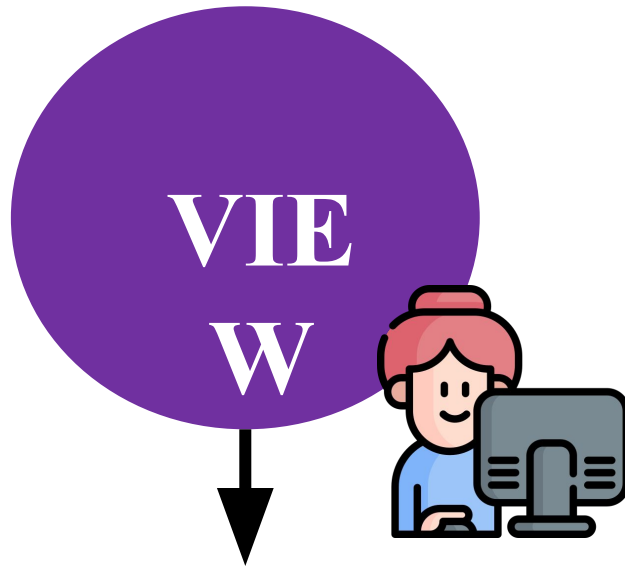
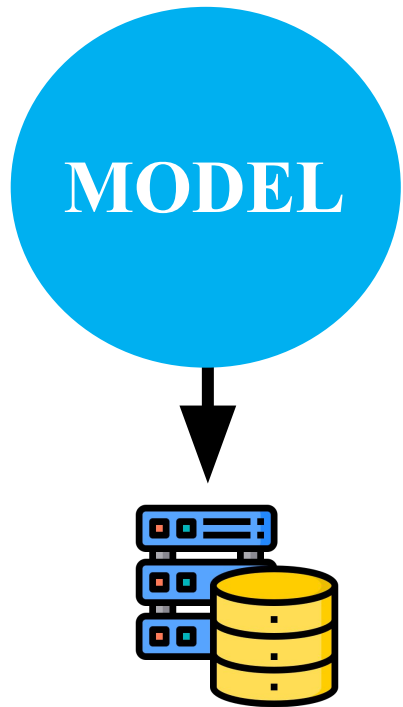
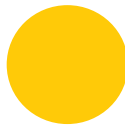


# MODEL-VIEW-CONTROLL



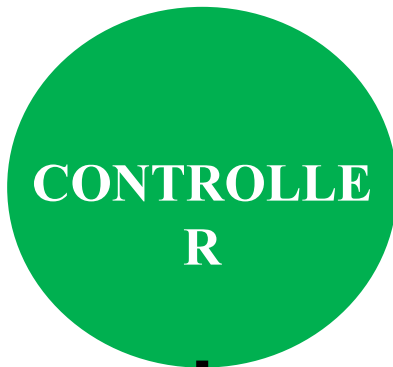
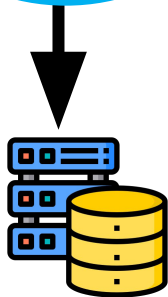
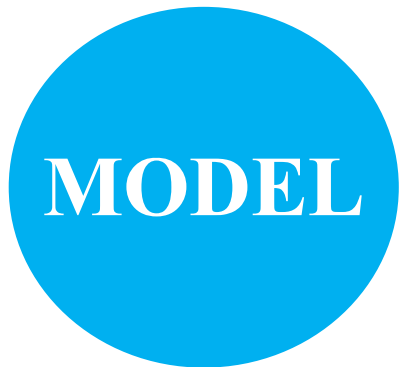
- Conectar com a fonte de dados
- Receber os dados
- Tratar os dados
- Validar os dados

# MODEL-VIEW-CONTROL



- Interface de comunicação com usuário
- Tabela, gráfico, formulário, uma página HTML
- Interação do usuário com o sistema

# MODEL-VIEW-CONTROLL ER

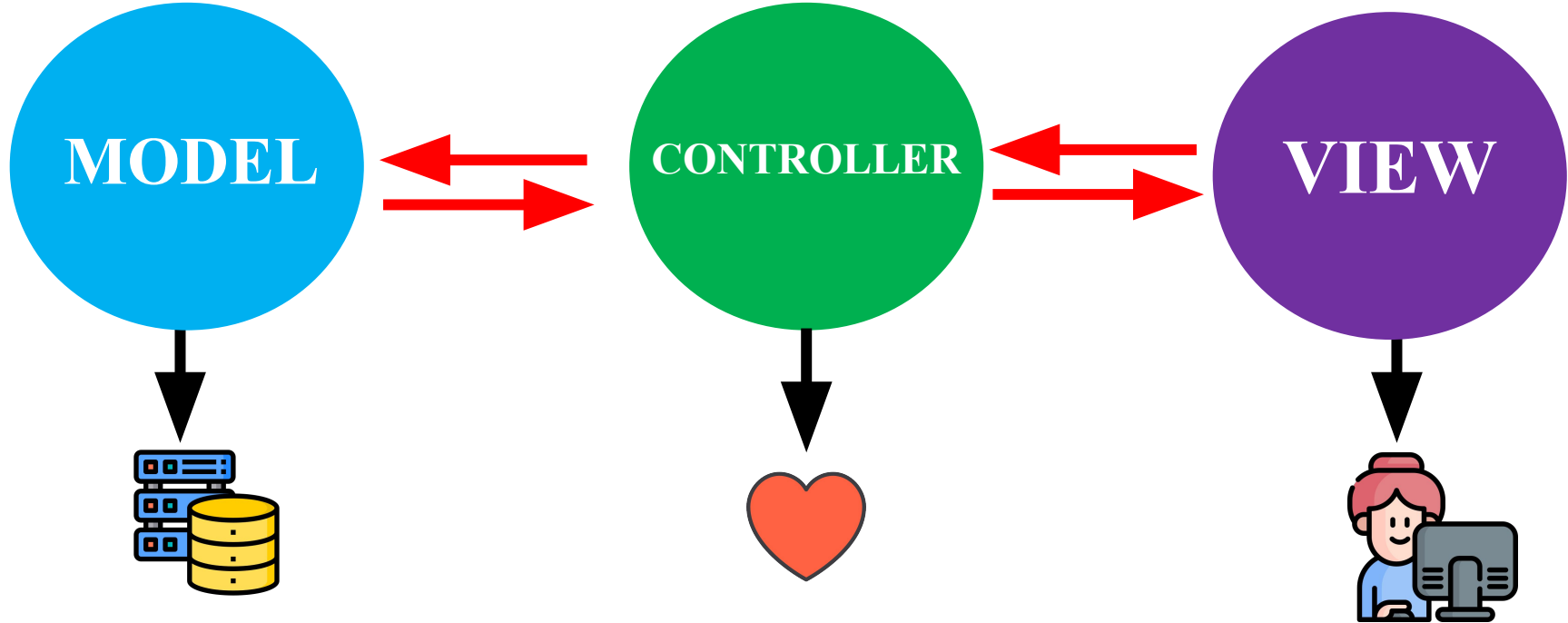


- Responsável por receber as requisições do usuário, trata-las e responde-las adequadamente
- Vai solicitar os dados do **model**
- Vai entregar os dados da **view**

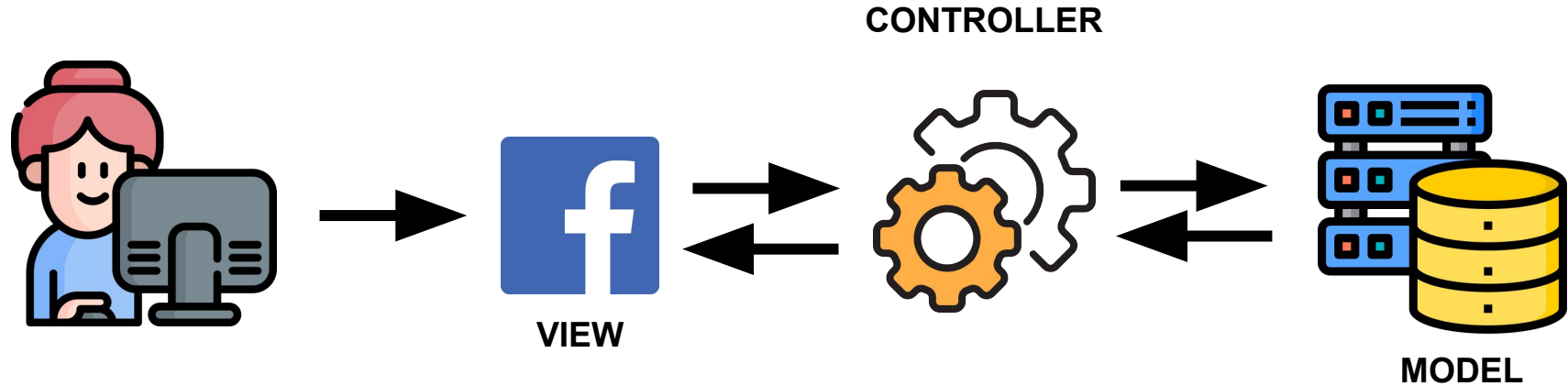




# MODEL-VIEW-CONTROLLER ER



# DIÁLOGO DAS CAMADAS



**View:** Fala Controller! O usuário acabou de pedir para acessar o Facebook! Pega os dados de login dele aí.

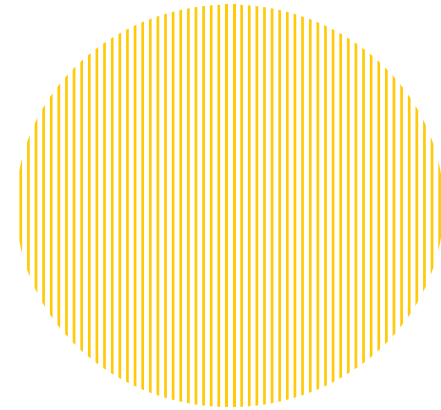
**Controller:** Blz. Já te mando a resposta. Ai model, meu parceiro, toma esses dados de login e verifica se ele loga.

**Model:** Os dados são válidos. Mandando a resposta de login.

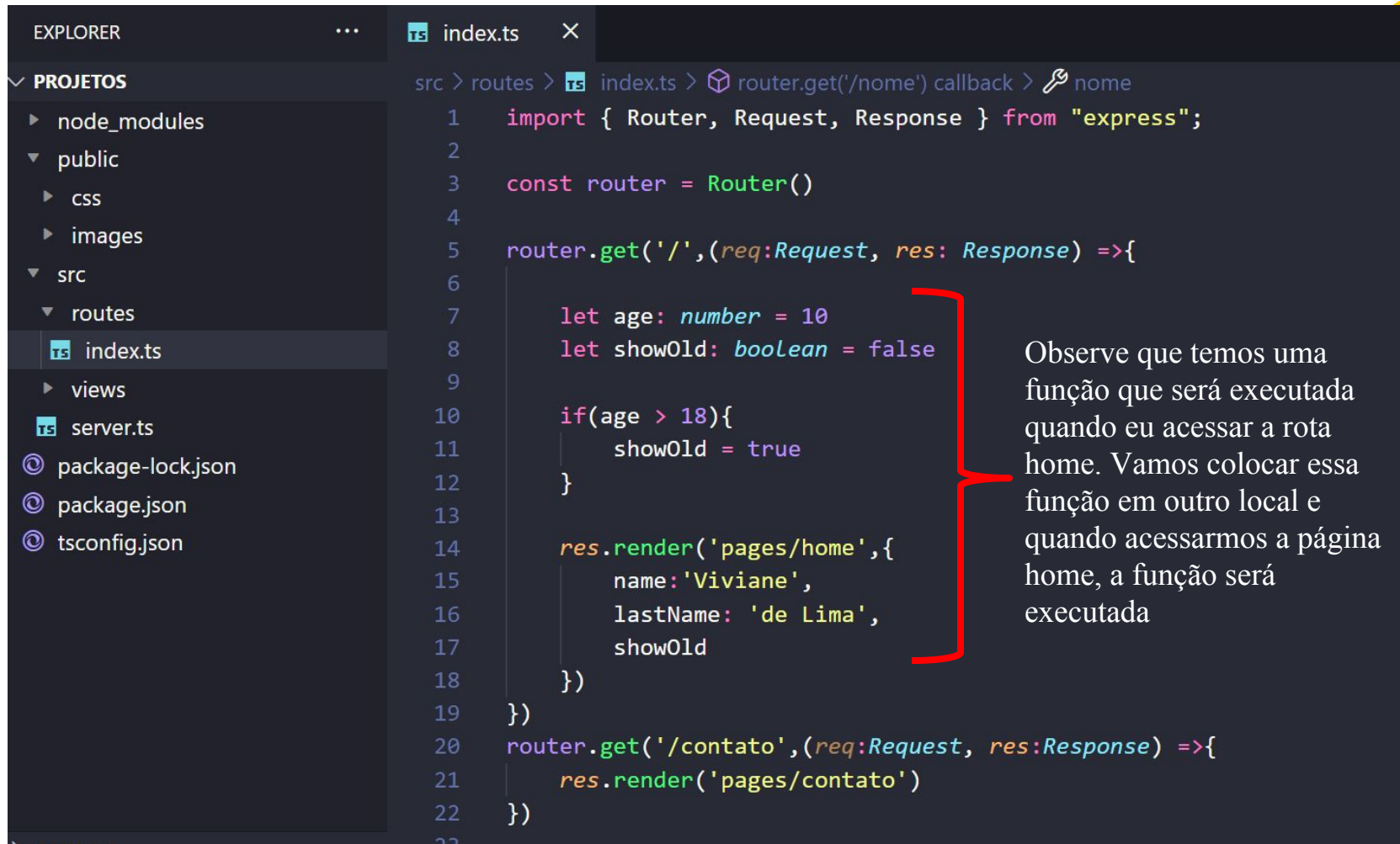
**Controller:** Blz, Model. View, o usuário informou os dados corretos. Vou mandar pra você os dados dele e você carrega a página de perfil.

**View:** Vlw. Mostrando ao usuário.

# **TRABALHANDO COM CONTROLLERS**



## VÁ EM INDEX.TS E OBSERVE O QUE TEMOS DENTRO DE HOME



EXPLORER

PROJETOS

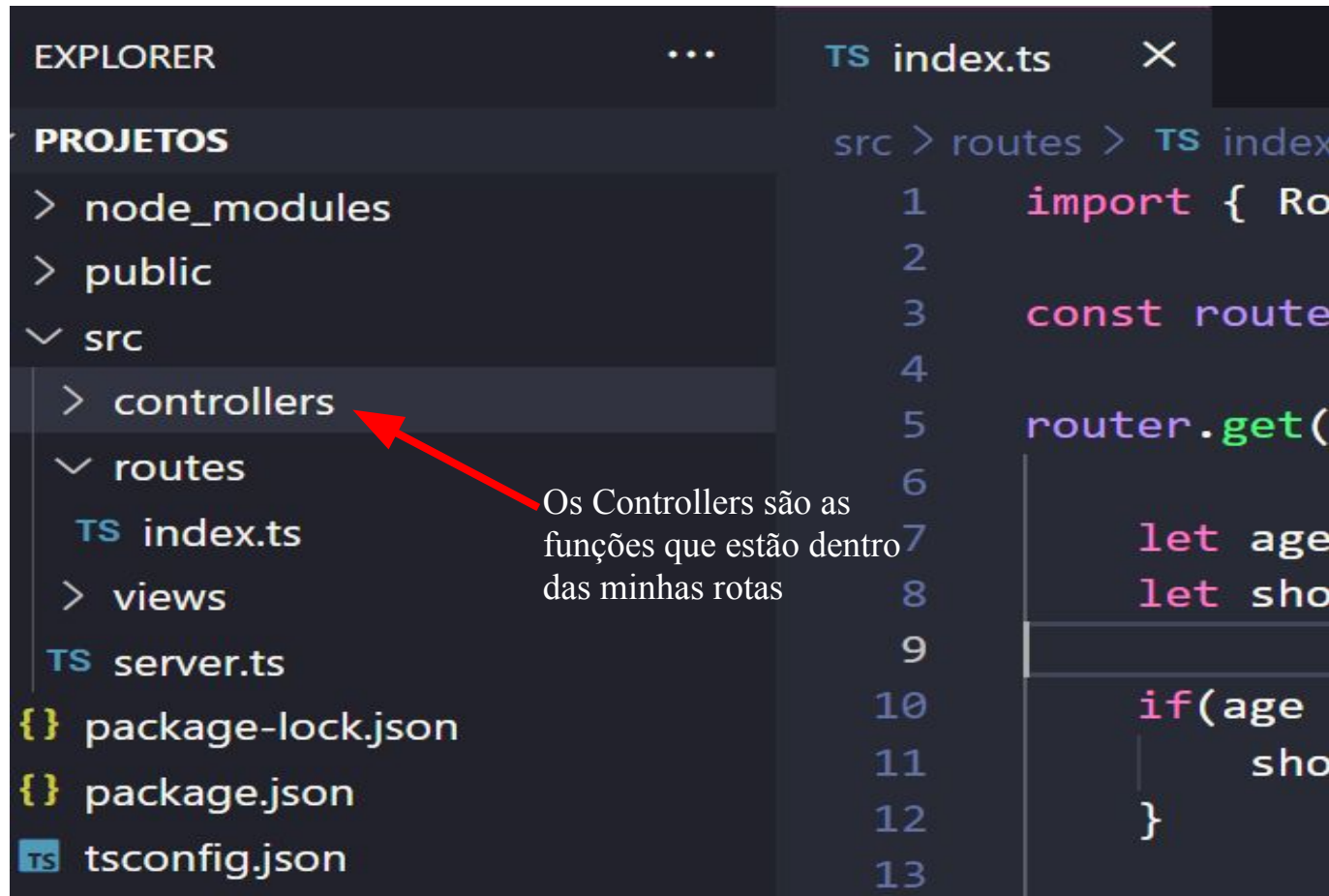
- node\_modules
- public
  - css
  - images
- src
  - routes
    - index.ts**
  - views
- server.ts
- package-lock.json
- package.json
- tsconfig.json

src > routes > **index.ts** > router.get('/nome') callback > nome

```
1 import { Router, Request, Response } from "express";
2
3 const router = Router()
4
5 router.get('/', (req: Request, res: Response) =>{
6
7     let age: number = 10
8     let showOld: boolean = false
9
10    if(age > 18){
11        showOld = true
12    }
13
14    res.render('pages/home',{
15        name: 'Viviane',
16        lastName: 'de Lima',
17        showOld
18    })
19 })
20 router.get('/contato', (req: Request, res: Response) =>{
21     res.render('pages/contato')
22 })
```

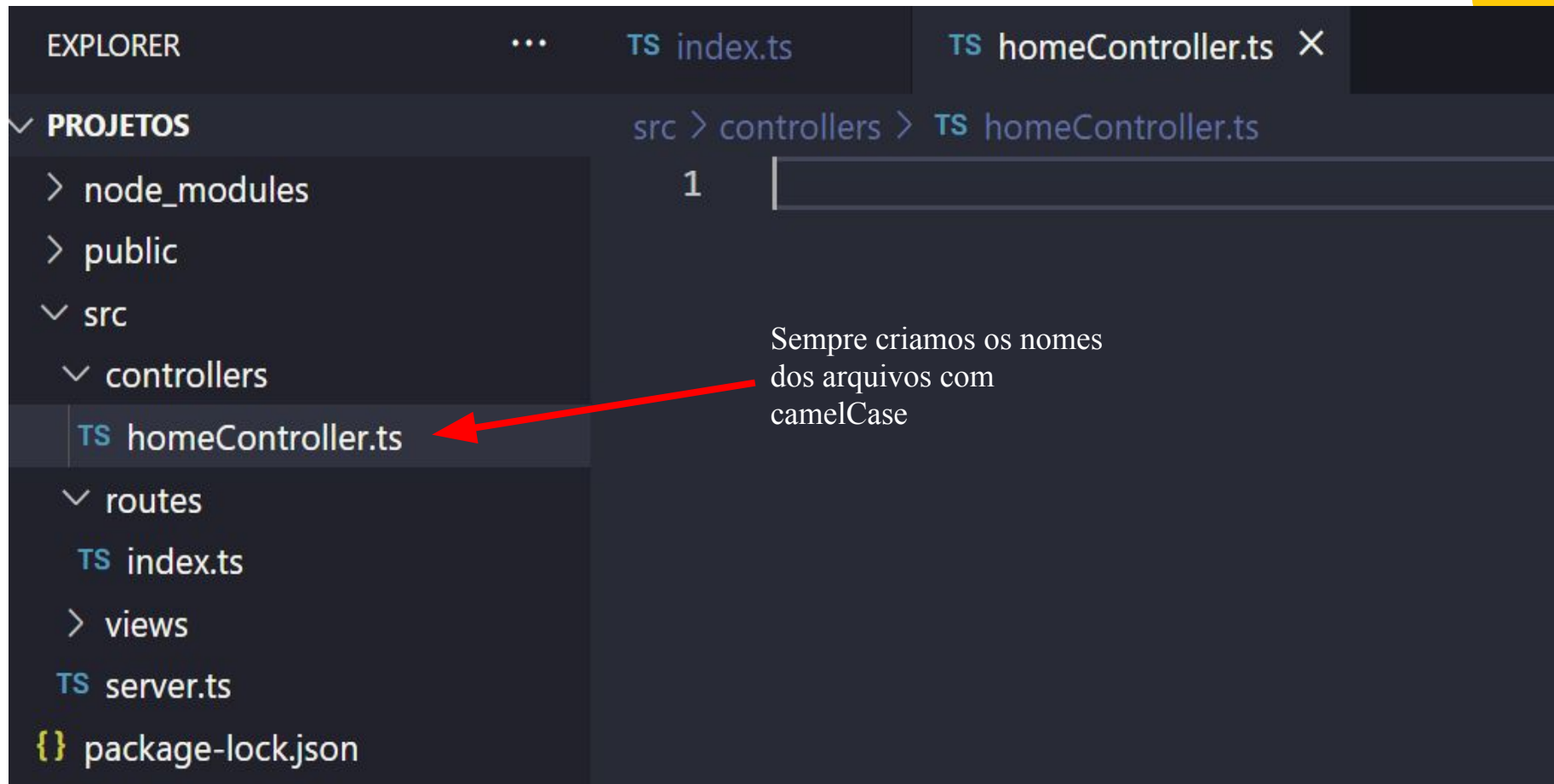
Observe que temos uma função que será executada quando eu acessar a rota home. Vamos colocar essa função em outro local e quando acessarmos a página home, a função será executada

## VÁ EM SRC E CRIE UMA PASTA NOVA CHAMADA CONTROLLERS



NO NOSSO ARQUIVO DE ROTAS VAMOS APENAS DEFINIR AS ROTAS E NADA MAIS

# CRIE O ARQUIVO HOMECONTROLLER.TS, ESSE ARQUIVO SERÁ RESPONSÁVEL PELOS CONTROLLERS DA NOSSA PÁGINA HOME



## COPIE A FUNÇÃO QUE ESTÁ NA SUA ROTA HOME (/) E COLE EM HOMECONTROLLER.TS

```
TS index.ts × TS homeController.ts ×
src > controllers > TS homeController.ts > ...
1  import {Request, Response} from 'express'
2
3  export const home = ((req:Request, res: Response) =>{
4
5      let age: number = 10
6      let showOld: boolean = false
7
8      if(age > 18){
9          showOld = true
10     }
11
12     res.render('pages/home',{
13         name:'Viviane',
14         lastName: 'de Lima',
15         showOld,
16         products: [
17             {title: 'produto X', price: 20},
18             {title: 'produto Y', price: 50},
19             {title: 'produto Z', price: 70},
20         ]
21     })
22 })
```

# NÃO ESQUEÇA DE EXPORTAR O HOME

EXPLORER

... TS index.ts TS homeController.ts X

✓ PROJETOS

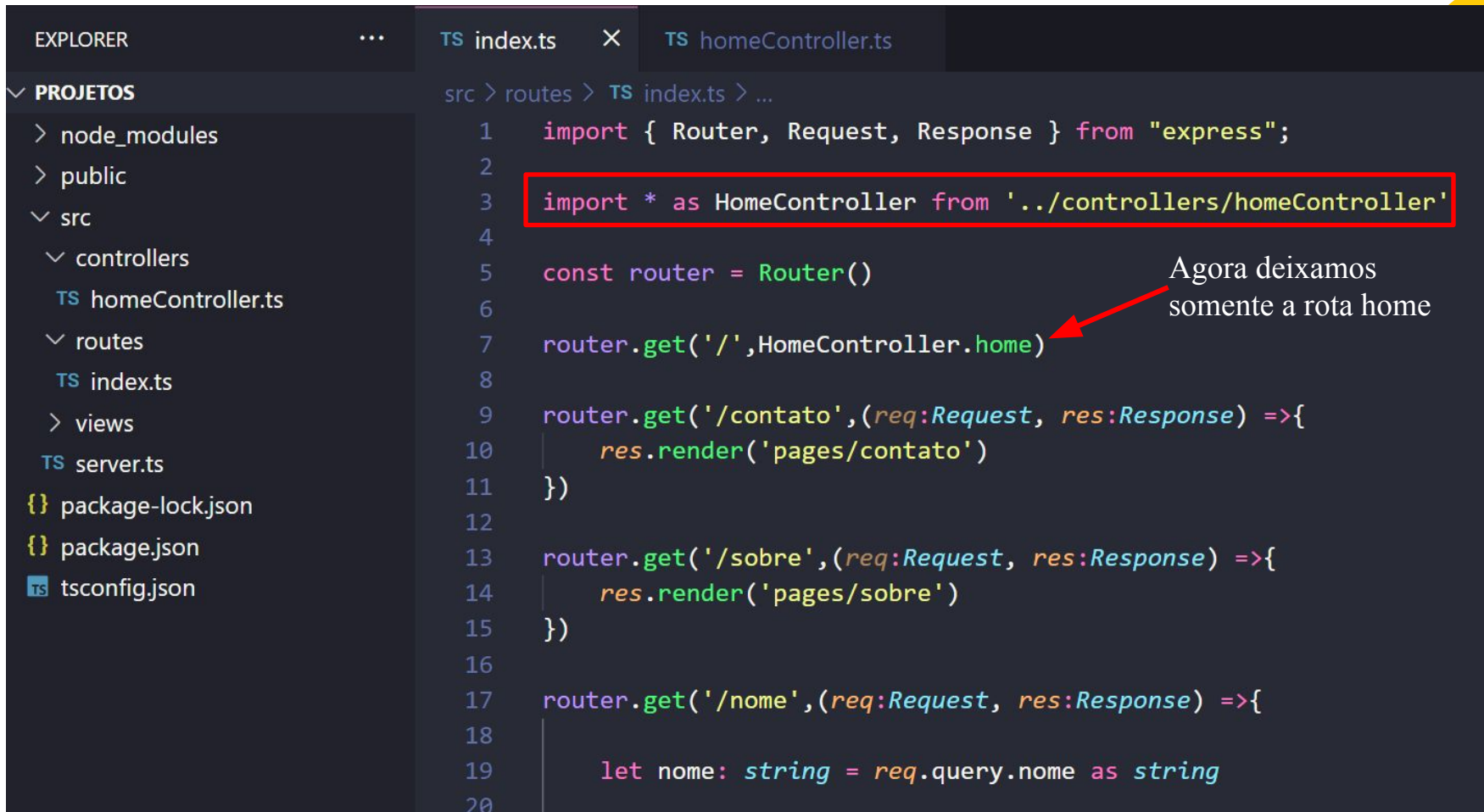
- > node\_modules
- > public
- ✓ src
  - ✓ controllers
    - TS homeController.ts
  - ✓ routes
    - TS index.ts
  - > views
- TS server.ts
- { } package-lock.json
- { } package.json
- TS tsconfig.json

src > controllers > TS homeController.ts > ...

```
1  import {Request, Response} from 'express'
2
3  export const home = ((req:Request, res: Response) =>{
4
5      let age: number = 10
6      let showOld: boolean = false
7
8      if(age > 18){
9          showOld = true
10     }
11
12     res.render('pages/home',{
13         name:'Viviane',
14         lastName: 'de Lima',
15         showOld
16     })
17 })
18
```



# IMPORTE A HOMECONTROLLER EM ROUTES>INDEX.TS



The image shows a VS Code editor window with two tabs: 'TS index.ts' and 'TS homeController.ts'. The left sidebar displays the file explorer with the following structure:

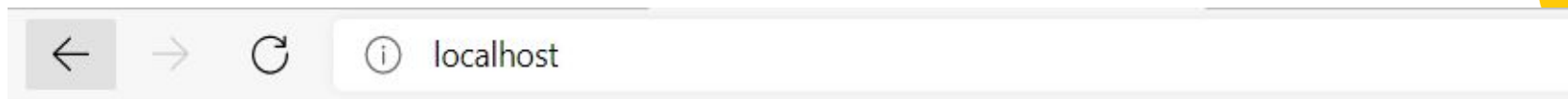
- PROJETOS
  - node\_modules
  - public
  - src
    - controllers
      - homeController.ts
    - routes
      - index.ts
    - views
  - server.ts
  - package-lock.json
  - package.json
  - tsconfig.json

The main editor area shows the content of 'src > routes > TS index.ts > ...':

```
1 import { Router, Request, Response } from "express";
2
3 import * as HomeController from '../controllers/homeController'
4
5 const router = Router()
6
7 router.get('/', HomeController.home)
8
9 router.get('/contato', (req: Request, res: Response) => {
10   res.render('pages/contato')
11 })
12
13 router.get('/sobre', (req: Request, res: Response) => {
14   res.render('pages/sobre')
15 })
16
17 router.get('/nome', (req: Request, res: Response) => {
18
19   let nome: string = req.query.nome as string
20 }
```

A red box highlights the import statement on line 3: `import * as HomeController from '../controllers/homeController'`. A red arrow points from the text 'Agora deixamos somente a rota home' to the `HomeController.home` property access on line 7.

**TESTE NO NAVEGADOR E VEJA SE A HOME AINDA ESTÁ FUNCIONANDO**



# Título da Página

---

## Produtos

---

Não temos frases motivacionais hoje

---

Todos os direitos reservados

**AGORA TENTE FAZER FAÇA O MESMO PROCEDIMENTO NAS OUTRAS  
ROTAS**

# INFOCONTROLLER.TS

EXPLORER

...

TS index.ts

TS homeController.ts

TS infoController.ts X

PROJETOS

node\_modules

public

src

controllers

TS homeController.ts

TS infoController.ts

routes

TS index.ts

views

pages

partials

TS server.ts

package-lock.json

package.json

tsconfig.json

src > controllers > TS infoController.ts > [X] sobre

1 import {Request,Response} from 'express'

2

3 export const contato = ((req:Request, res:Response) =>{

4 | res.render('pages/contato')

5 |})

6

7 export const sobre = ((req:Request, res:Response) =>{

8 | res.render('pages/sobre')

9 |})

# INDEX.TS

EXPLORER

...

TS index.ts X TS homeController.ts TS infoController.ts

PROJETOS

node\_modules

public

src

controllers

homeController.ts

infoController.ts

routes

index.ts

views

pages

partials

server.ts

package-lock.json

package.json

src > routes > TS index.ts > ...

```
1 import { Router, Request, Response } from "express";
2
3 import * as HomeController from '../controllers/homeController'
4 import * as infoController from '../controllers/infoController'
5
6 const router = Router()
7
8 router.get('/',HomeController.home)
9 router.get('/contato',infoController.contato)
10 router.get('/sobre',infoController.sobre)
11
12 router.get('/:nome',(req:Request, res:Response) =>{
13
14     let nome: string = req.query.nome as string
15
16     res.render('pages/nome',{
```

# USERCONTROLLER.TS

TS index.ts

TS userController.ts X

src > controllers > TS userController.ts > [⌕] idadeAction

```
1  import { Request, Response } from "express";
2
3  export const nome = (req:Request, res:Response) =>{
4
5      let nome: string = req.query.nome as string
6      let idade: string = req.query.idade as string
7
8      res.render('pages/nome',{
9          nome,
10         idade
11     })
12 }
13 export const idadeForm = (req:Request, res:Response) =>{
14     res.render('pages/idade')
15 }
16 export const idadeAction = (req:Request, res:Response) =>{
17     let mostrarIdade: boolean = false
18     let idade: number = 0
19     if(req.body.anoNascimento){
20         let anoNascimento: number = parseInt(req.body.anoNascimento as string)
21         let anoAtual: number = new Date().getFullYear()
22         idade = anoAtual - anoNascimento
23         mostrarIdade = true
24     }
25
26     res.render('pages/idade',{
27         idade,
28         mostrarIdade
29     })
30 }
```



# INDEX.TS

EXPLORER

...

PROJETOS

> node\_modules

> public

> src

> controllers

TS homeController.ts

TS infoController.ts

TS userController.ts

> routes

TS index.ts

> views

> pages

> partials

TS server.ts

{ } package-lock.json

{ } package.json

src > routes > TS index.ts > ...

1 import { Router, Request, Response } from "express";

2

3 import \* as HomeController from '../controllers/homeController'

4 import \* as infoController from '../controllers/infoController'

5 import \* as userController from '../controllers/userController'

6

7 const router = Router()

8

9 router.get('/',HomeController.home)

10 router.get('/contato',infoController.contato)

11 router.get('/sobre',infoController.sobre)

12 router.get('/nome',userController.nome)

13 router.get('/idade',userController.idadeForm)

14 router.post('/idade-resultado',userController.idadeAction)

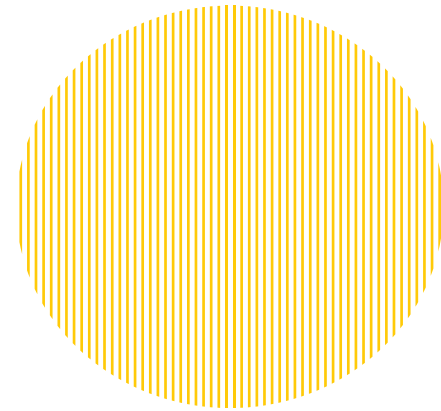
15

16

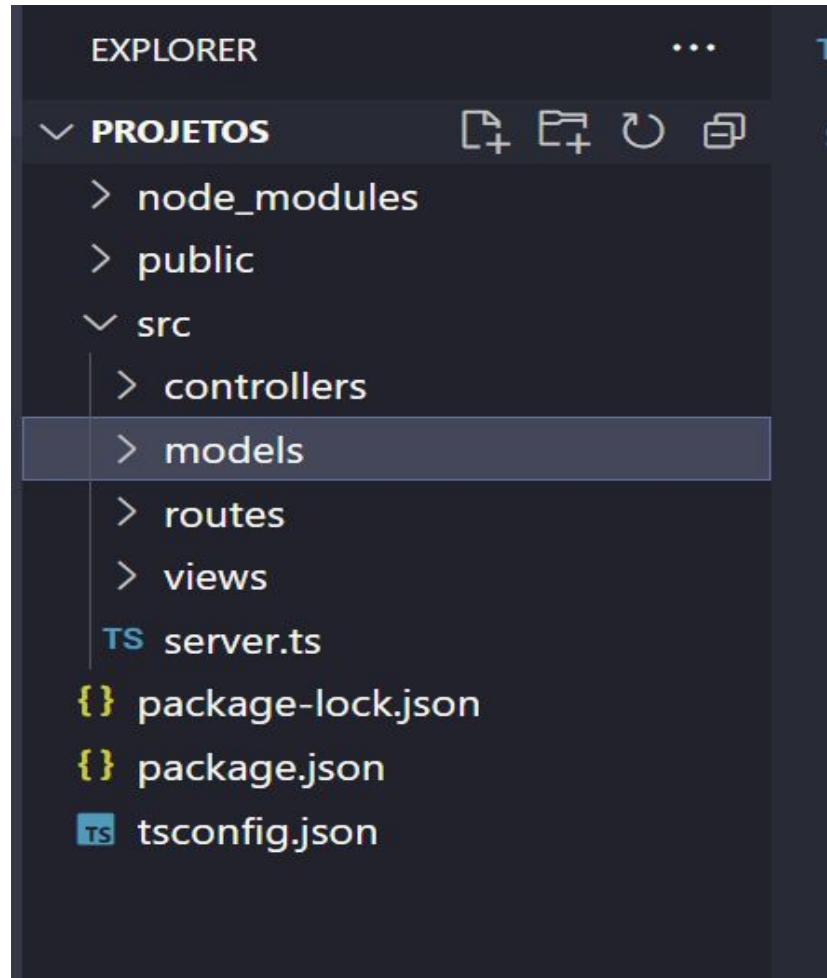
17 export default router

18

# **TRABALHANDO COM MODELS**



# CRIE UMA PASTA CHAMADA MODELS DENTRO DE SRC





## VAMOS USAR OS PRODUTOS QUE ESTÃO NA NOSSA HOMECONTROLLER COMO

TS index.ts × TS homeController.ts ×

src > controllers > TS homeController.ts > ...

```
1  import {Request, Response} from 'express'
2
3  export const home = ((req:Request, res: Response) =>{
4
5      let age: number = 10
6      let showOld: boolean = false
7
8      if(age > 18){
9          showOld = true
10     }
11
12     res.render('pages/home',{
13         name:'Viviane',
14         lastName: 'de Lima',
15         showOld,
16         products: [
17             {title: 'produto X', price: 20},
18             {title: 'produto Y', price: 50},
19             {title: 'produto Z', price: 70},
20         ]
21     })
22 })
```

Os produtos de um site não devem ficar de qualquer jeito como vemos aqui em homeController.ts, nesse caso se faz necessário criar os models

**VAMOS FINGIR QUE ESTAMOS CRIANDO UMA LOJA VIRTUAL E VAMOS USAR O  
MODEL PARA EXIBIR UM TIPO DE ITEM ESPECÍFICO**



# Título da Página

---

## Produtos

- produto X - R\$ 20
- produto Y - R\$ 50
- produto Z - R\$ 70

---

Não temos frases motivacionais hoje

---

Todos os direitos reservados

**EXEMPLO: MEU SITE É UM PETSHOP COM CADASTRO DE USUÁRIOS, VOU TER UM  
MODEL PARA O USUÁRIO E UM MODEL PARA AS INFORMAÇÕES RELACIONADAS AO  
PET**

# COMO ESTAMOS USANDO O EXEMPLO DE PRODUTOS, CRIEI UM MODEL PARA

OS



```
> node_modules
> public
✓ src
  ✓ controllers
    TS homeController.ts
    TS infoController.ts
    TS userController.ts
  ✓ models
    TS Product.ts
  ✓ routes
    TS index.ts
  > views
    TS server.ts
  {} package-lock.json
  {} package.json
  TS tsconfig.json
```

**PARA CRIAR ARQUIVOS NO MODEL SE ATENTE  
AS REGRAS:**

**1 - NOME SEMPRE NO SINGULAR (POIS  
PODEMOS TRATAR DE UM ITEM OU VÁRIOS  
ITEMS)**

**2 – USAMOS SEMPRE PASCAL CASE (PRIMEIRA  
LETRA E SEGUNDA SEMPRE MAIUSCULA)**

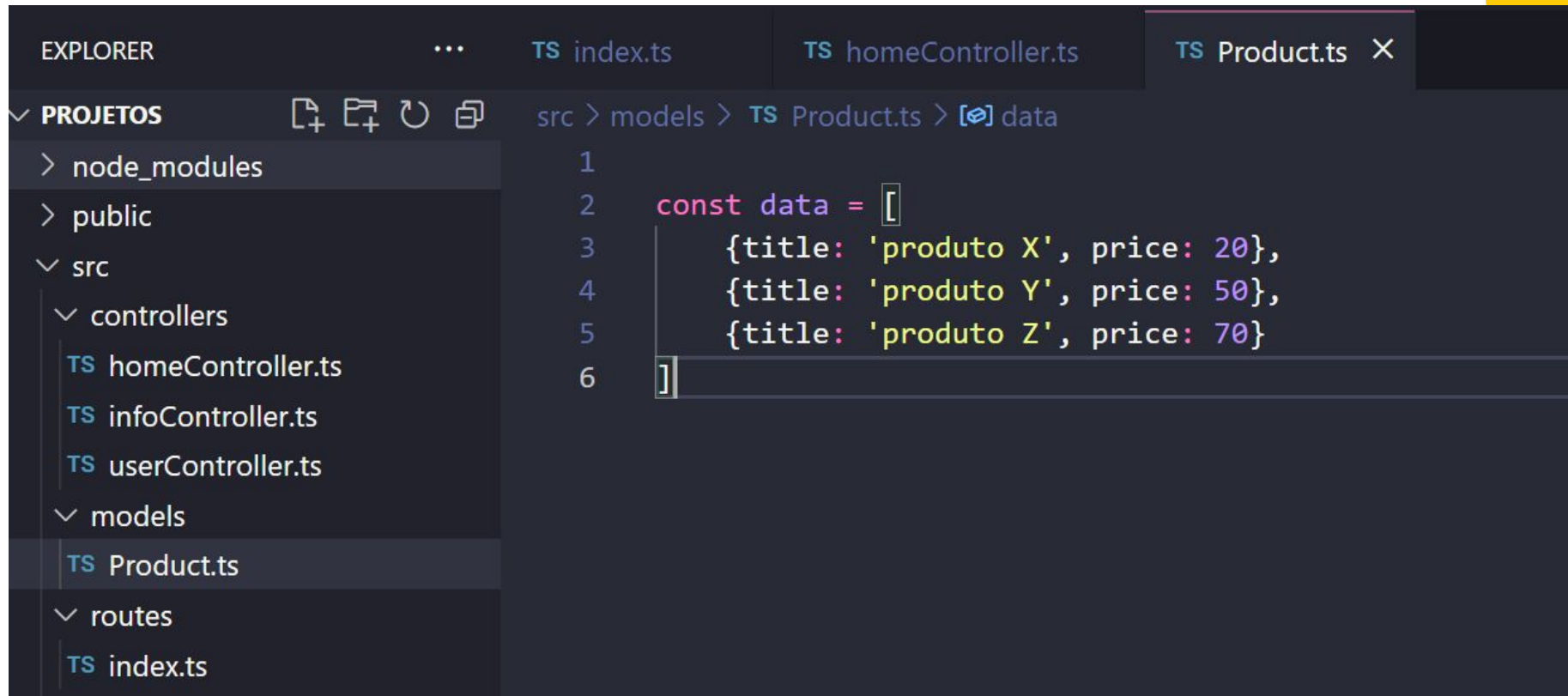
**EM PRODUCTS VAMOS CRIAR AS FUNÇÕES NECESSÁRIAS PARA  
QUE O SISTEMA FUNCIONE**



**MAS QUE TIPO DE FUNÇÃO?**

- **CRIAR UMA FUNÇÃO QUE PEGUE TODOS OS MEUS  
PRODUTOS DO BANCO DE DADOS**
- **CRIAR UMA FUNÇÃO QUE SEJA POSSÍVEL PESQUISAR O  
PRODUTO PELO NOME**

## COMO AINDA NÃO VIMOS BANCO DE DADOS COM NODEJS, VAMOS CRIAR UMA VARIÁVEL DE EXEMPLO E VAMOS USÁ-LA COMO SE FOSSE NOSSO BANCO DE DADOS



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 like 'node\_modules', 'public', and 'src'. Under 'src', there are subfolders 'controllers', 'models', and 'routes'. The 'models' folder is expanded, and 'Product.ts' is selected. The main editor area shows the content of 'Product.ts', which is a TypeScript file. The code defines a constant array named 'data' containing three objects, each representing a product with a 'title' and a 'price'.

```
1  
2  const data = [  
3    {title: 'produto X', price: 20},  
4    {title: 'produto Y', price: 50},  
5    {title: 'produto Z', price: 70}  
6  ]
```

ENTRETANTO SE DEIXARMOS ASSIM O TYPESCRIPT NÃO VAI CURTIR MUITO ESSE FORMATO, ENTÃO PRECISAMOS TIPAR OS PRODUTOS QUE ESTÃO DENTRO DE DATA

## SE O MEU PRODUTO EM TIPO E PREÇO, PRECISAMOS INFORMAR AO TYPESCRIPT

TS index.ts

TS homeController.ts

TS Product.ts X

src > models > TS Product.ts > ...

```
1  //tipando o tittle e o price com Typescript
2  type Product = {
3      tittle:string,
4      price:number,
5  }
6
7  const data = [
8      {title: 'produto X', price: 20},
9      {title: 'produto Y', price: 50},
10     {title: 'produto Z', price: 70}
11 ]
12
```

# CRIANDO UMA FUNÇÃO DE EXEMPLO



TS index.ts

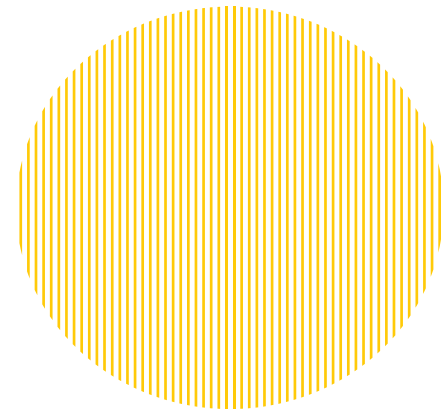
TS homeController.ts

TS Product.ts X

src > models > TS Product.ts > ...

```
1  type Product = {
2    tittle:string,
3    price:number,
4  }
5  const data = [
6    {title: 'produto X', price: 20},
7    {title: 'produto Y', price: 50},
8    {title: 'produto Z', price: 70}
9  ]
10
11  /*Em seguida vamos criar as funções
12  vamos criar uma função que pegue todos os produtos
13  do nosso banco de dados*/
14  export const Product = {
15    getAll: () => {
16      //aqui dentro colocamos a função
17      //que vai retornar o nosso banco DATA
18      return data
19    }
20  }
```

# **VARIÁVEIS DE AMBIENTE**



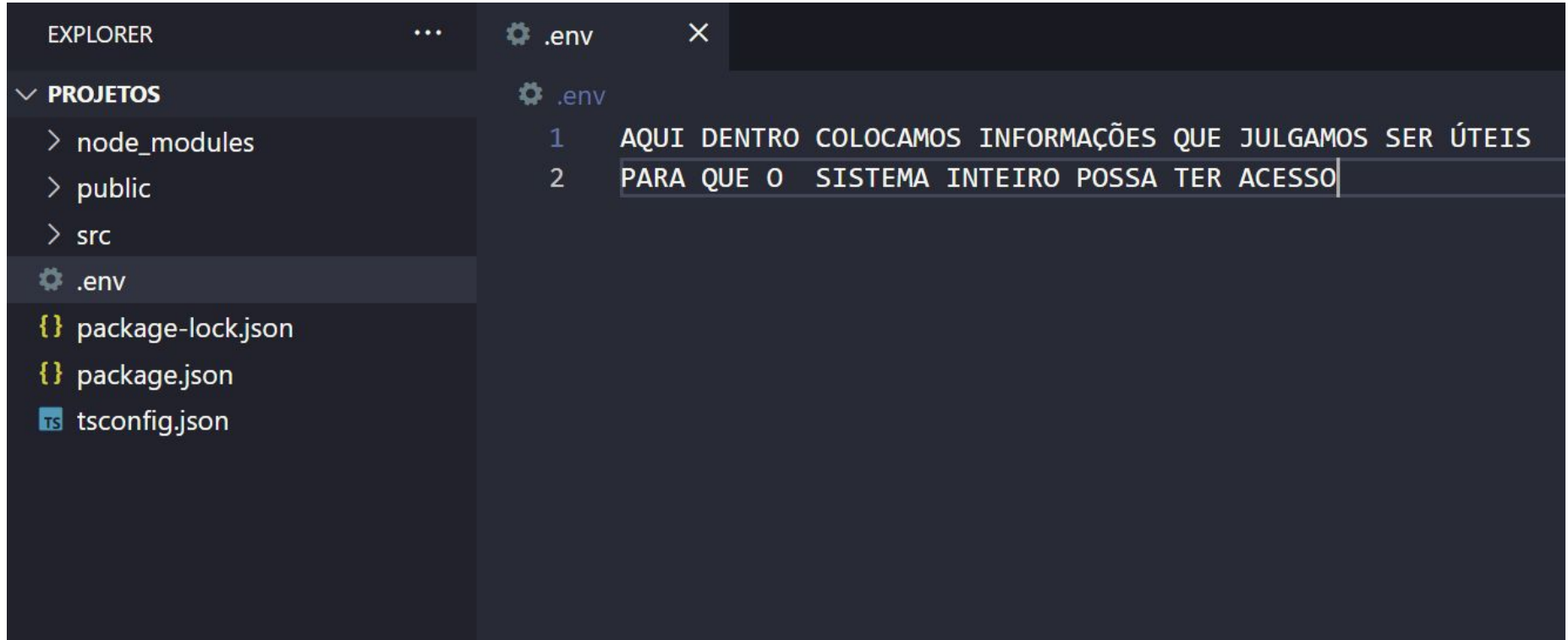


## O QUE SÃO VARIÁVEIS DE AMBIENTE?



Uma variável de ambiente é um valor dinâmico que o sistema operacional e outro software podem usar para determinar informações específicas para o seu computador. Em outras palavras, uma variável de ambiente é algo que representa outra coisa, como uma localização em seu computador, um número de versão, uma lista de objetos. Existem **variáveis de ambiente do usuário** e **variáveis de ambiente do sistema**

# CRIE UM NOVO ARQUIVO CHAMADO .ENV NA RAÍZ DO SEU PROJETO (ENV SIGNIFICA ENVIRONMENT)

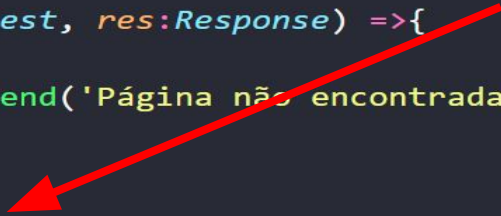


# VAMOS SUPOR QUE TEMOS VÁRIOS SISTEMAS FUNCIONANDO, CADA SISTEMA PRECISA RODAR EM UMA PORTA DIFERENTE

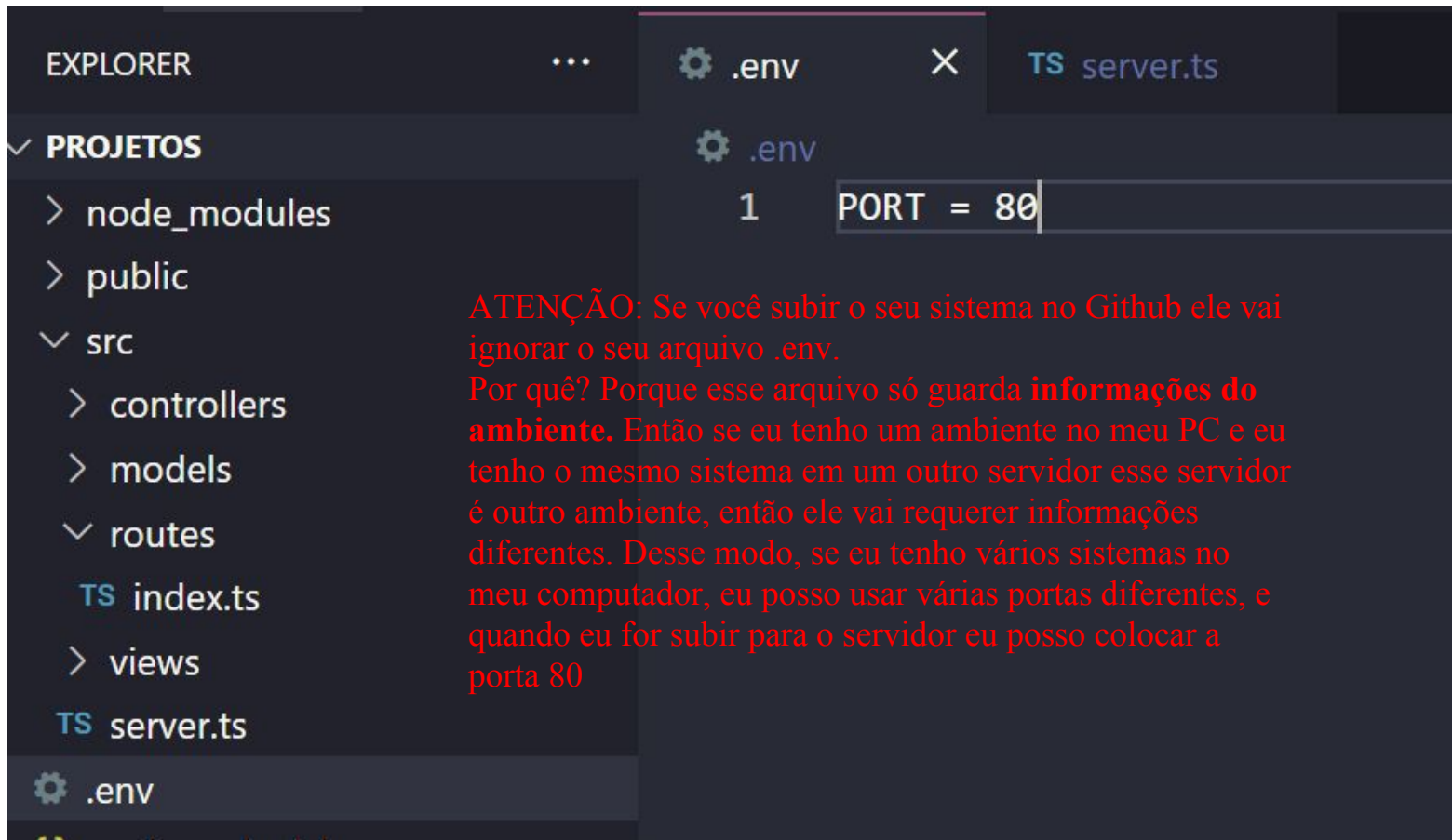


```
.env TS server.ts X
src > TS server.ts > ...
1  import express,{Request, Response} from 'express'
2  import mustache from 'mustache-express'
3  import path from 'path'
4  import mainRoutes from './routes/index'
5
6  const server = express()
7
8  server.set('view engine','mustache')
9  server.set('views',path.join(__dirname,'views'))
10 server.engine('mustache',mustache())
11 server.use(express.static(path.join(__dirname,'../public')))
12
13 server.use(express.urlencoded({extended:true}))
14
15 server.use(mainRoutes)
16
17 server.use((req: Request, res:Response) =>{
18
19     res.status(404).send('Página não encontrada!')
20 })
21
22 server.listen(3000)
23
```

Mesmo que nosso servidor esteja na porta 3000,  
quando colocarmos nosso sistema no ar  
automaticamente ele vai rodar na porta 80



# ENTÃO PRECISAMOS INFORMAR AO NOSSO ARQUIVO .ENV QUAL É A PORTA QUE NOSSO SISTEMA VAI RODAR



The screenshot shows the VS Code interface. On the left, the Explorer sidebar is open, showing a project structure with folders like 'node\_modules', 'public', 'src', 'controllers', 'models', 'routes', and 'views'. The 'src' folder is expanded, showing files like 'index.ts' and 'server.ts'. The 'server.ts' file is selected. On the right, the '.env' file is open, showing the line 'PORT = 80'. A red text box is overlaid on the right side of the image, containing a warning in Portuguese.

ATENÇÃO: Se você subir o seu sistema no Github ele vai ignorar o seu arquivo .env.  
Por quê? Porque esse arquivo só guarda **informações do ambiente**. Então se eu tenho um ambiente no meu PC e eu tenho o mesmo sistema em um outro servidor esse servidor é outro ambiente, então ele vai requerer informações diferentes. Desse modo, se eu tenho vários sistemas no meu computador, eu posso usar várias portas diferentes, e quando eu for subir para o servidor eu posso colocar a porta 80

# NO SEU ARQUIVO .ENV COLOQUE A PORTA 4000



The image is a screenshot of the Visual Studio Code editor interface. On the left, the Explorer sidebar shows a project structure under the heading 'PROJETOS'. The folders 'node\_modules', 'public', and 'src' are expanded. Inside 'src', there are subfolders 'controllers', 'models', and 'routes', and files 'index.ts', 'server.ts', and 'views'. The file 'server.ts' is selected. Below the Explorer, the file explorer shows '.env', 'package-lock.json', 'package.json', and 'tsconfig.json'. The '.env' file is selected and its content is displayed in the main editor area. The content of '.env' is 'PORT = 4000'. The text 'Como vamos conseguir usar essa porta dentro do meu sistema?' is overlaid on the editor area.

EXPLORER

PROJETOS

- > node\_modules
- > public
- src
  - > controllers
  - > models
  - routes
    - TS index.ts
  - > views
  - TS server.ts

.env

1 PORT = 4000

Como vamos conseguir usar essa porta dentro do meu sistema?

# INSTALE A BIBLIOTECA DOTENV NO SEU TERMINAL



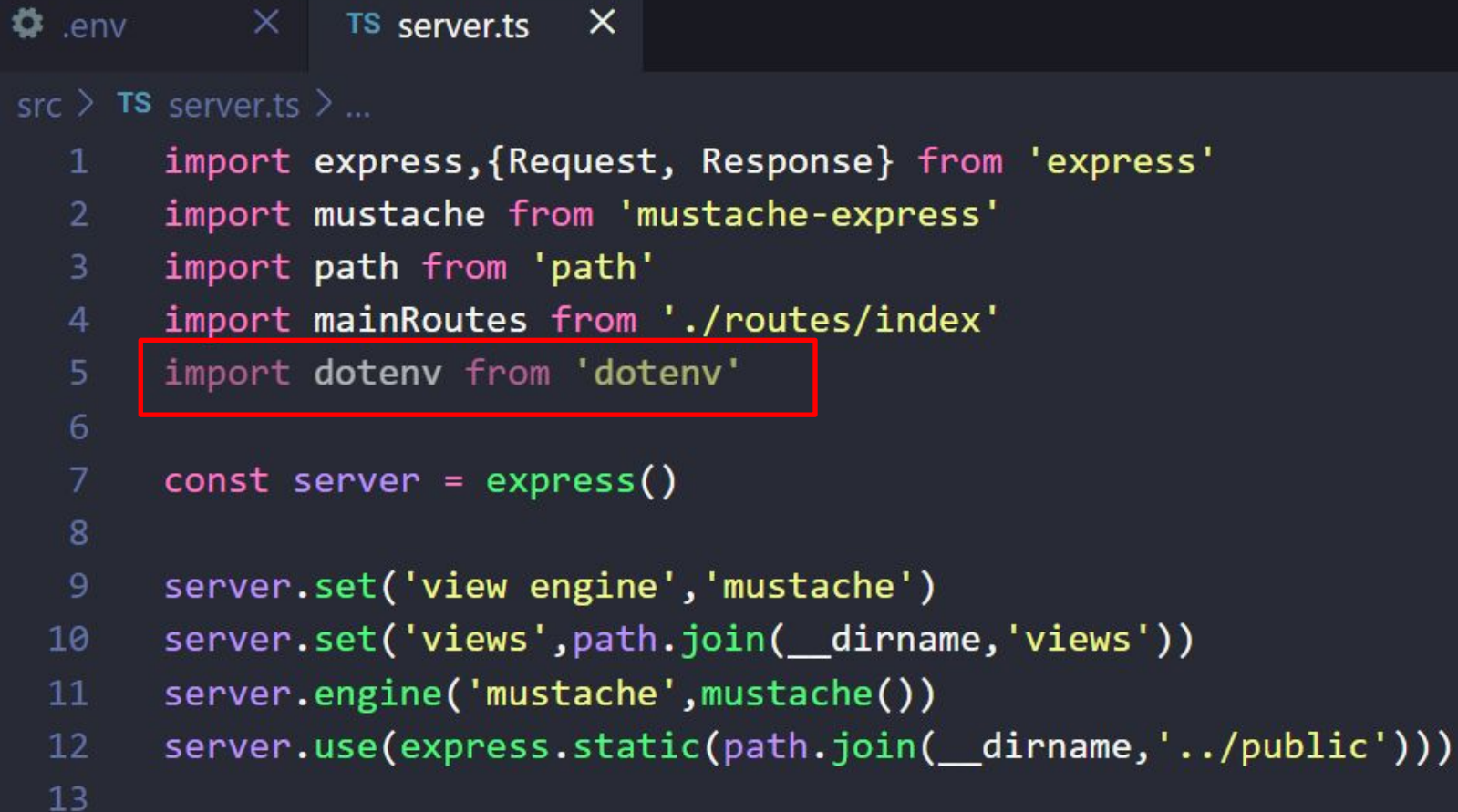
The image shows a Visual Studio Code editor window with a dark theme. At the top, there are two tabs: '.env' (with a gear icon) and 'TS server.ts'. The '.env' tab is active, showing a single line of code: 'PORT = 4000'. Below the editor, there is a panel with four tabs: 'PROBLEMS', 'OUTPUT', 'TERMINAL' (which is selected and underlined), and 'DEBUG CONSOLE'. The 'TERMINAL' tab displays the output of a command executed in a terminal. The text in the terminal is as follows:

```
PS C:\Users\Viviane\Downloads\node\projetos> npm install dotenv
added 1 package, and audited 71 packages in 2s

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

found 0 vulnerabilities
```

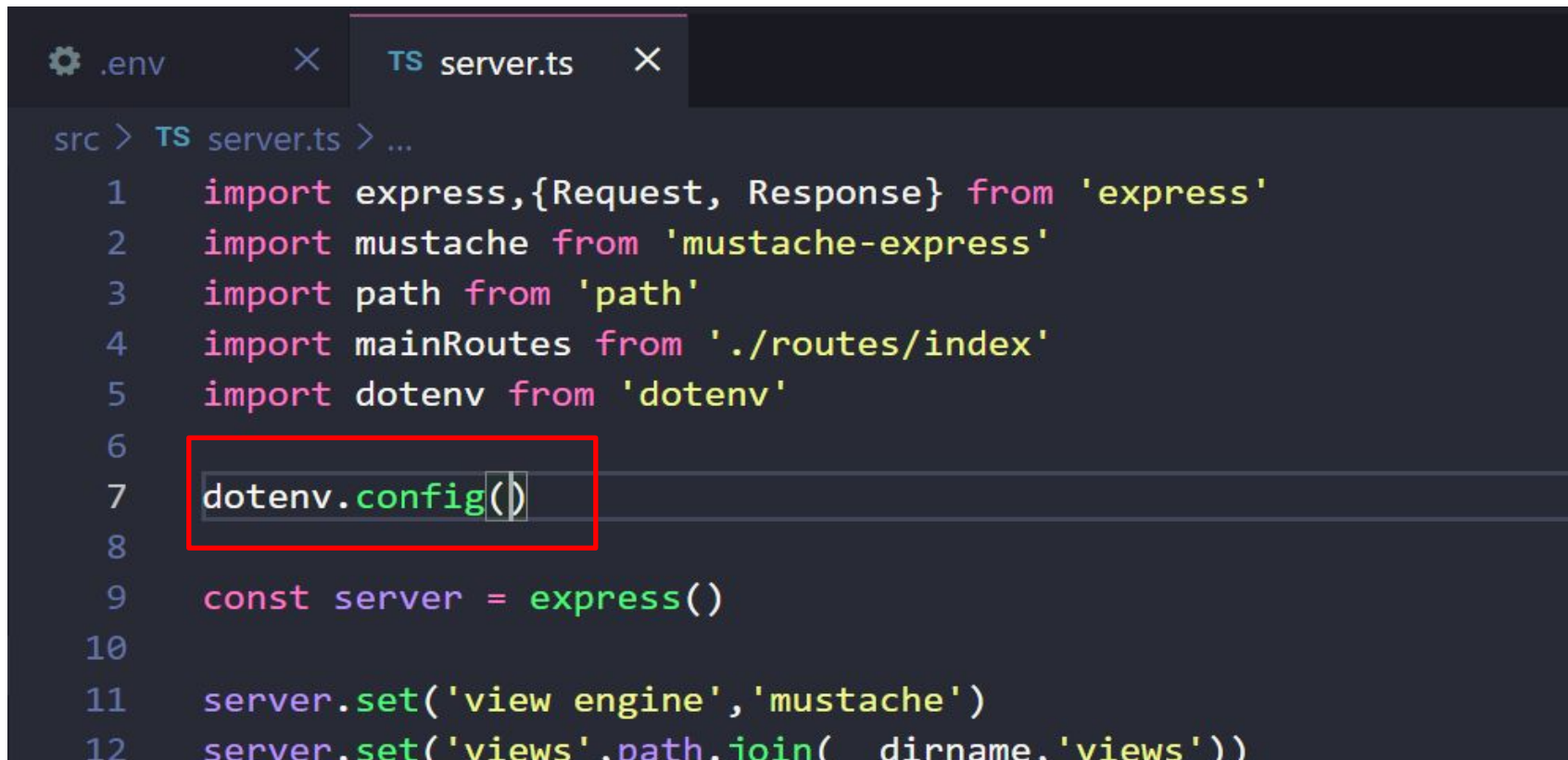
## VÁ EM SERVER.TS E IMPORTE O DOTENV



The image shows a code editor with two tabs: `.env` and `TS server.ts`. The `TS server.ts` tab is active, showing a TypeScript file. The code imports several modules: `express`, `mustache`, `path`, and `mainRoutes`. The line `import dotenv from 'dotenv'` is highlighted with a red rectangle. Below the imports, the code sets up an Express server with Mustache as the view engine and a static file directory at `../public`.

```
src > TS server.ts > ...
1  import express,{Request, Response} from 'express'
2  import mustache from 'mustache-express'
3  import path from 'path'
4  import mainRoutes from './routes/index'
5  import dotenv from 'dotenv'
6
7  const server = express()
8
9  server.set('view engine','mustache')
10 server.set('views',path.join(__dirname,'views'))
11 server.engine('mustache',mustache())
12 server.use(express.static(path.join(__dirname,'../public')))
13
```

## RODE A FUNÇÃO CONFIG DO DOTENV



The image shows a VS Code editor window with two tabs: `.env` and `TS server.ts`. The `server.ts` file is open, showing the following code:

```
src > TS server.ts > ...  
1  import express,{Request, Response} from 'express'  
2  import mustache from 'mustache-express'  
3  import path from 'path'  
4  import mainRoutes from './routes/index'  
5  import dotenv from 'dotenv'  
6  
7  dotenv.config()  
8  
9  const server = express()  
10  
11  server.set('view engine','mustache')  
12  server.set('views',path.join(__dirname,'views'))
```

The `dotenv.config()` line on line 7 is highlighted with a red rectangle, indicating the function being executed to load environment variables.

AGORA NOSSO SISTEMA TEM ACESSO AS VARIÁVEIS DE AMBIENTE



## AGORA VAMOS USAR A PORTA QUE COLOCAMOS NO .ENV NO NOSSO SERVIDOR



```
src > TS server.ts > ...  
11  server.set('view engine','mustache')  
12  server.set('views',path.join(__dirname,'views'))  
13  server.engine('mustache',mustache())  
14  server.use(express.static(path.join(__dirname,'../public')))  
15  
16  server.use(express.urlencoded({extended:true}))  
17  
18  server.use(mainRoutes)  
19  
20  server.use((req: Request, res:Response) =>{  
21      |  
22      |     res.status(404).send('Página não encontrada!')  
23      | })  
24  
25  //PORT É O NOME QUE COLOCAMOS PARA NOSSA VARIÁVEL LÁ EM .ENV  
26  server.listen(process.env.PORT)
```



← → ↻ ⓘ localhost:4000

## Título da Página

---

### Produtos

- produto X - R\$ 20
- produto Y - R\$ 50
- produto Z - R\$ 70

---

Não temos frases motivacionais hoje

---

Todos os direitos reservados

**NUNCA MAIS ESQUEÇA: INFORMAÇÕES QUE VÃO MUDAR DE AMBIENTE PARA  
AMBIENTE COLOQUE EM .ENV**