

Desenvolvendo WebServices

1. A instalação do ambiente de desenvolvimento

1.1. Plataforma de desenvolvimento

Como plataforma de desenvolvimento, vamos utilizar o Framework **Node.js** que é uma solução para desenvolvimento do **lado servidor** que utiliza a linguagem de programação **JavaScript**.

Principal vantagem do Node.js é o fato de ela ser altamente escalável e extremamente leve. Tem uma comunidade grande e muito ativa, que disponibiliza um grande número de bibliotecas e extensões.

É gratuito, e suas aplicações são geradas utilizando a Licença MIT.

Link para download: <https://nodejs.org/en/>.

1.2. Gerenciador de Pacotes

O gerenciador de pacotes, vai nos auxiliar nos downloads e inclusão de bibliotecas de terceiros que porventura podemos precisar, e também vai nos auxiliar na publicação dos nossas APIs que serão construídas.

Vamos optar por **Yarn** e quem tiver interesse, procurem sobre Yarn x NPM.

Link para download: <https://classic.yarnpkg.com/en/docs/install#windows-stable>

1.3. IDE de Desenvolvimento

Vamos adotar o Visual Studio Code (VS Code) que é uma ferramenta gratuita da Microsoft para desenvolvimento nas mais diversas linguagens.

Link para download: <https://code.visualstudio.com/>

2. Testando o ambiente de desenvolvimento

→ Acesse a área de pesquisa do windows



→ digite no prompt de comando **node -v**

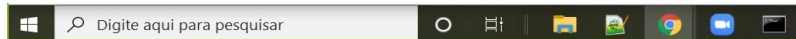
⇒ caso apareça a mensagem: **node não é reconhecido como uma comando interno**, fazer o download do node (<https://nodejs.org/en/>) e instalá-lo no computador.

→ digite no prompt de comando **yarn -v**

⇒ caso apareça a mensagem: **yarn não é reconhecido como uma comando interno**, fazer o download do yarn (<https://classic.yarnpkg.com/en/docs/install#windows-stable>) e instalá-lo no computador.

3. Criar um diretório para os códigos fonte.

→ Acesse a área de pesquisa do windows



→ digite **cmd** e escolher o aplicativo **prompt de comando**.

→ digite no prompt de comando **cd ** para ir para a raiz da pasta **C:\>**

→ digite no prompt de comando **mkdir workspaces** para criar uma pasta chamada projetos.

→ digite no prompt de comando **cd workspaces** para entrar na pasta chamada projetos. **C:\workspaces>**

→ digite no prompt de comando **mkdir assti** para criar uma pasta chamada projetos.

→ digite no prompt de comando **cd assti** para entrar na pasta chamada projetos. **C:\workspaces\assti>**

→ digite no prompt de comando **mkdir pratica01** para criar uma pasta chamada pratica01.

→ digite no prompt de comando **cd pratica01** para entrar na pasta chamada projetos. **C:\projetos\assti\pratica01>**

4. Criar o meu projeto

Para a criação do projeto da prática 01, iremos utilizar o gerenciador de pacotes **yarn** através do comando **yarn init -y**

C:\projetos\assti\pratica01> yarn init -y

5. Abrir o VSCode para criar meus códigos fontes

Para abrir o VSCode, basta digitar o comando **code** . no prompt de comandos do Windows.

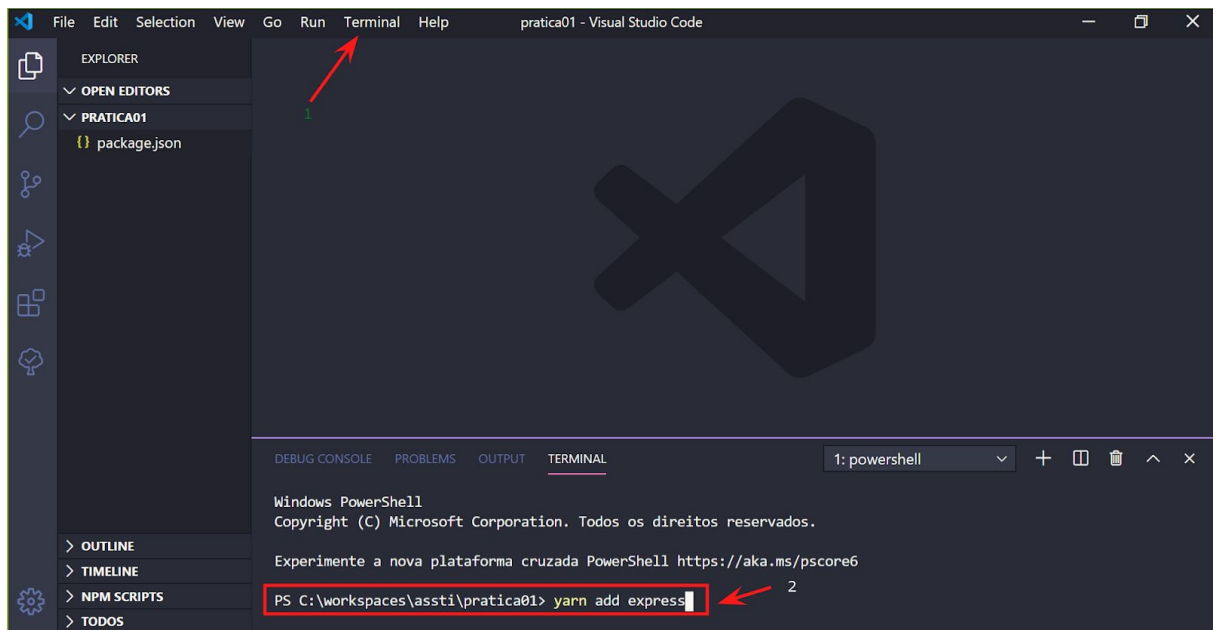
C:\projetos\assti\pratica01> code .

6. Adicionar o framework express ao nosso projeto

Abrir o Terminal no VSCode, indo no menu Terminal ⇒ New Terminal

E digitar no terminal do VSCode o comando **yarn add express**.

A documentação do express se encontra em: <https://expressjs.com/pt-br/>



7. Criar minha API de WebServices que realiza um CRUD de alunos.

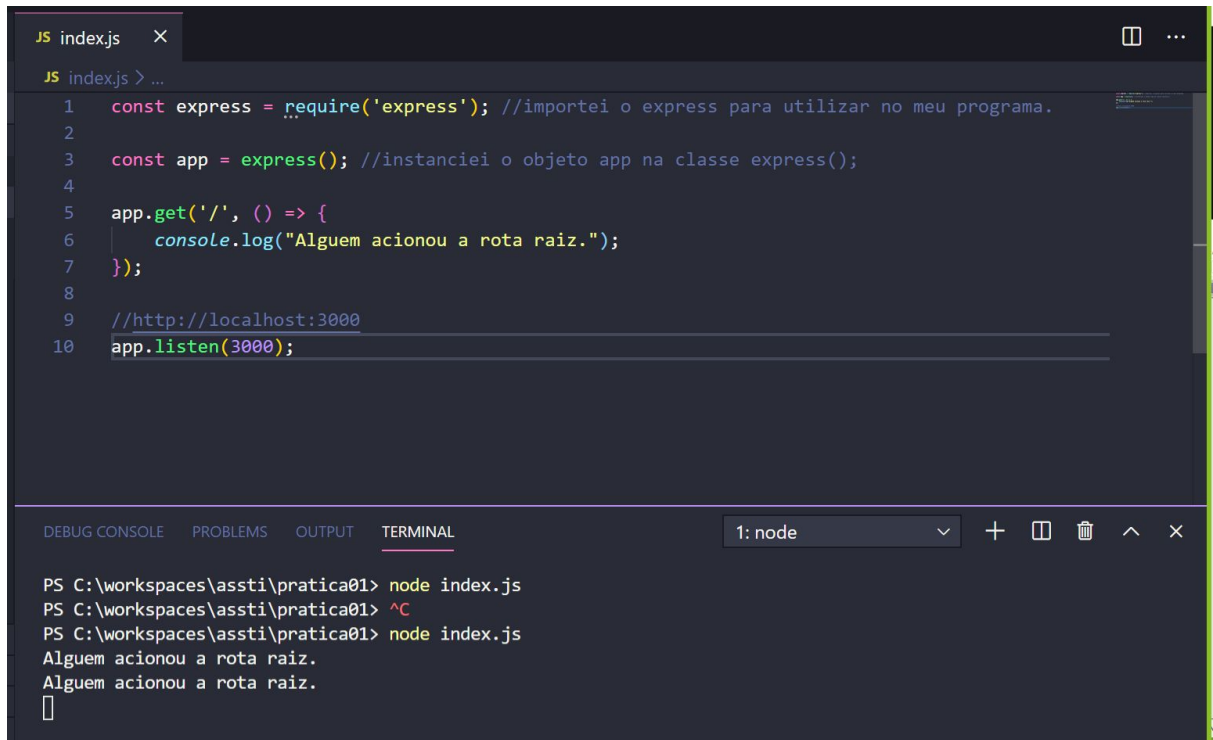
Passo 1: Criar o arquivo chamado **index.js** e nele codificar conforme o exemplo abaixo, que é responsável por criar um servidor do tipo http que responde na porta 3000.



Passo 2: testar se nosso código fonte funciona corretamente. Basta digitar no terminal do VSCode o comando **node index.js**.

Passo 3: Abrir o navegador e digitar no endereço **<http://localhost:3000>**.

Passo 4: É definir as rotas da nossa API. As rotas, são as chamadas de cada um dos serviços que nossa API possui. Sempre utilizam o mesmo formato: **metodo uri**, onde um exemplo de rota é **GET <http://localhost:3000/>**



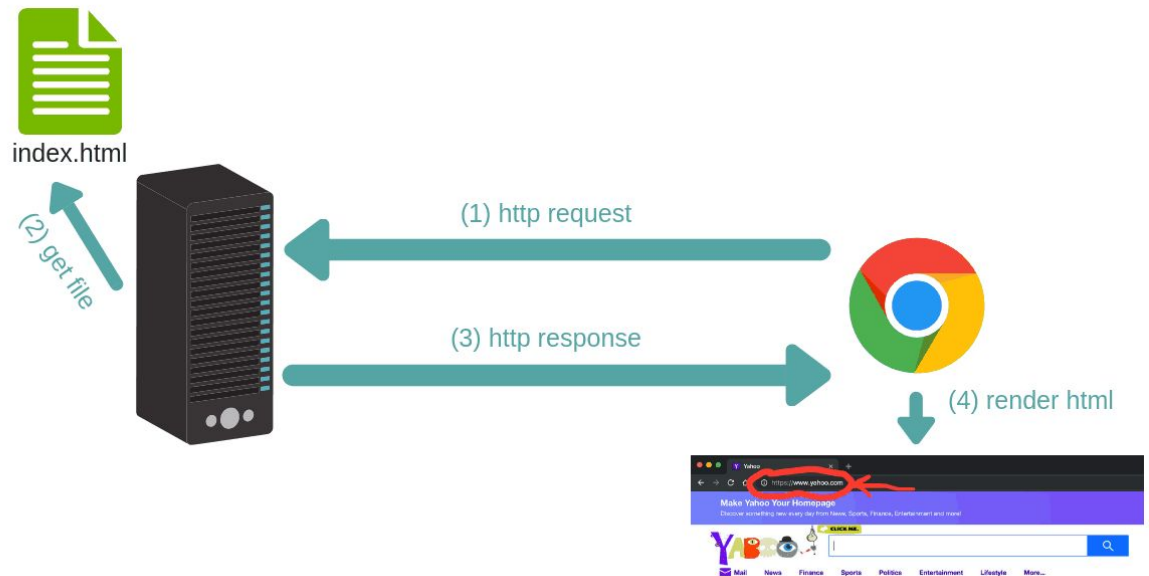
```
JS index.js X
JS index.js > ...
1  const express = require('express'); //importei o express para utilizar no meu programa.
2
3  const app = express(); //instanciei o objeto app na classe express();
4
5  app.get('/', () => {
6    console.log("Alguem acionou a rota raiz.");
7  });
8
9  //http://localhost:3000
10 app.listen(3000);

DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL
1: node
PS C:\workspaces\assti\pratica01> node index.js
PS C:\workspaces\assti\pratica01> ^C
PS C:\workspaces\assti\pratica01> node index.js
Alguem acionou a rota raiz.
Alguem acionou a rota raiz.
[]
```

Passo 5: testar se nosso código fonte funciona corretamente. Basta digitar no terminal do VSCode o comando **node index.js**.

Passo 6: Abrir o navegador e digitar no endereço **<http://localhost:3000>**. Se no terminal do VSCode aparecer o texto “Alguem acionou a rota raiz” é porque o meu código da funcionando.

Passo 7: adequar o nosso código para funcionar conforme um serviço do tipo http, que deve ser baseado em uma requisição e uma resposta, conforme ilustra a imagem a seguir.



Passo 8: Para adequar o nosso código, devemos incluir dois **parâmetros** na nossa função anônima criada em nossa rota raiz do tipo get, os parâmetros são **req** (que representa a requisição) e **res** (que representa a resposta).

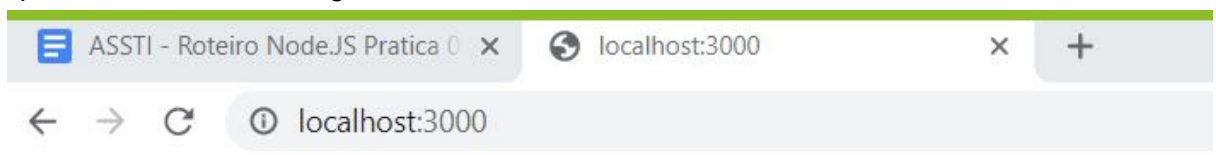
```
JS index.js •
JS index.js > ...
1  const express = require('express'); //importei o express para utilizar no meu programa.
2
3  const app = express(); //instanciei o objeto app na classe express();
4
5  app.get('/', (req, res) => {
6    console.log("Alguem acionou a rota raiz.");
7  });
8
9  //http://localhost:3000
10 app.listen(3000);
```

Passo 9: Além dos parâmetros, minha função anônima tem que **retornar** na **resposta** o que eu desejo que seja visualizado no cliente (navegador).

```
JS index.js  X
JS index.js > ...
1  const express = require('express'); //importei o express para utilizar no meu programa.
2
3  const app = express(); //instanciei o objeto app na classe express();
4
5  app.get('/', (req, res) => {
6      console.log("Alguem acionou a rota raiz.");
7      return res.send("Servidor esta funcionando corretamente.");
8  });
9
10 //http://localhost:3000
11 app.listen(3000);
```

Passo 10: testar se nosso código fonte funciona corretamente. Digitar primeiro **Ctrl + C** na finalizar uma execução anterior do node, caso ela exista e depois basta digitar no terminal do VSCode o comando **node index.js**.

Passo 11: Abrir o navegador e digitar no endereço <http://localhost:3000>. Se aparecer conforme a imagem abaixo, esta funcionando OK.



Servidor esta funcionando corretamente.

Construindo nossa primeira API.

Na aula de hoje, vamos construir uma API de **cadastro de pessoas**, sem conexão em banco de dados, apenas para entendermos os conceitos básicos associados a APIs e WebService.

Em uma **API**, é muito comum termos as operações de um **CRUD** (Create, Read, Update e Delete), que são as 4 operações transacionais de banco de dados.

Operação	BD Relacional	Rota API METODO uri	Método no Controlador
Create	insert	POST localhost:3000/pessoas	store
Read	select	GET localhost:3000/pessoas GET localhost:3000/pessoas/id	index show

Update	update	PUT localhost:3000/pessoas	update
Delete	delete	DELETE localhost:3000/pessoas	destroy

É que normalmente, um Webservice REST retorna a informação no formato JSon.

Abrindo o Projeto no VSCode

→ Acesse a área de pesquisa do windows



→ digite **cmd** e escolher o aplicativo **prompt de comando**.

→ digite no prompt de comando **cd ** para ir para a raiz da pasta **C:\>**

→ digite no prompt de comando **cd workspaces** para entrar na pasta chamada projetos. **C:\workspaces>**

→ digite no prompt de comando **cd assti** para entrar na pasta chamada projetos. **C:\workspaces\assti>**

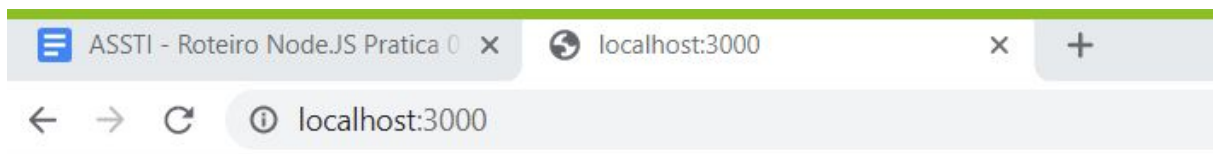
→ digite no prompt de comando **cd pratica01** para entrar na pasta chamada projetos. **C:\projetos\assti\pratica01>**

→ digite no prompt de comando **code .** para abrir o projeto da pasta atual. **C:\projetos\assti\pratica01> code .**

Testando o Projeto

Passo 1: testar se nosso código fonte funciona corretamente. Basta digitar no terminal do VSCode o comando **node index.js**.

Passo 2: Abrir o navegador e digitar no endereço <http://localhost:3000>. Se aparecer conforme a imagem abaixo, esta funcionando OK.

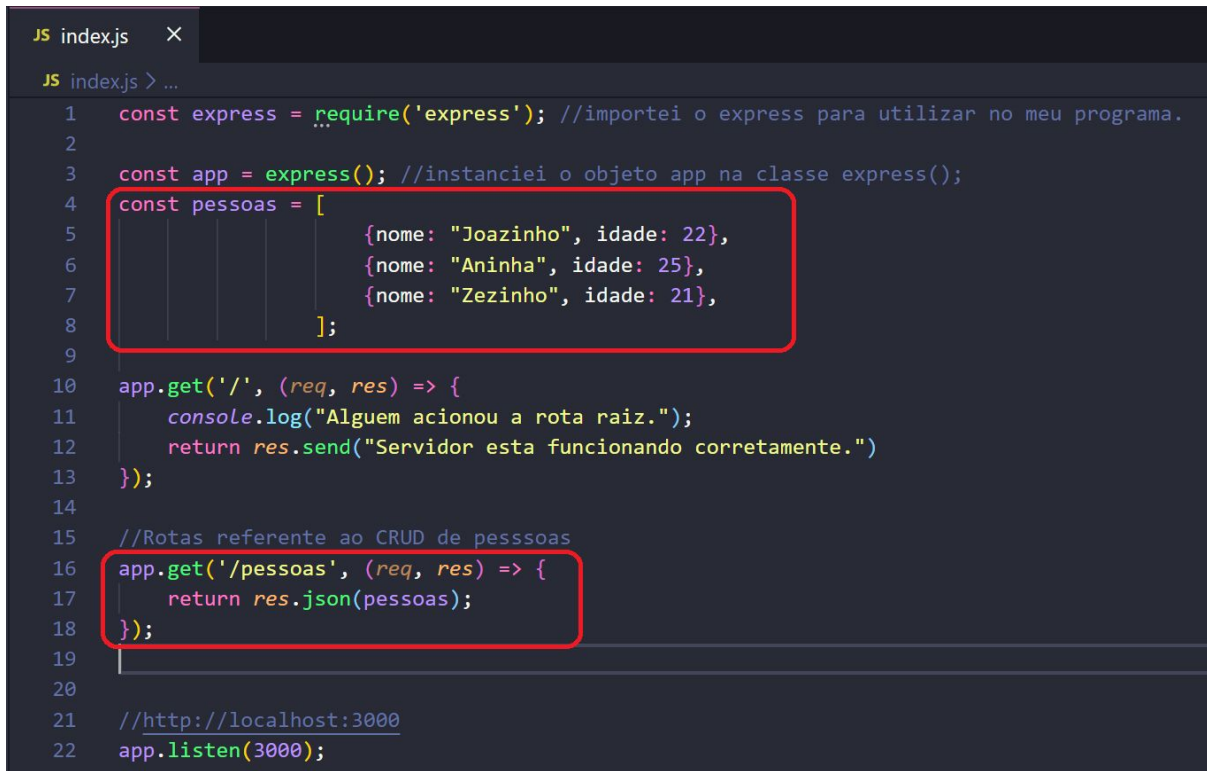


Servidor esta funcionando corretamente.

Implementando a primeira Rota: **GET** localhost:3000/pessoas

Este endpoint é responsável por listar todas as pessoas cadastradas no serviço de pessoas. Como ainda não estamos conectando em Banco de Dados, criamos um vetor para representar as pessoas cadastradas.

Passo 1: Alterar o arquivo **index.js** conforme a figura abaixo.



```
JS index.js  X
JS index.js > ...
1  const express = require('express'); //importei o express para utilizar no meu programa.
2
3  const app = express(); //instanciei o objeto app na classe express();
4  const pessoas = [
5      {nome: "Joazinho", idade: 22},
6      {nome: "Aninha", idade: 25},
7      {nome: "Zezinho", idade: 21},
8  ];
9
10 app.get('/', (req, res) => {
11     console.log("Alguem acionou a rota raiz.");
12     return res.send("Servidor esta funcionando corretamente.")
13 });
14
15 //Rotas referente ao CRUD de pesssoas
16 app.get('/pessoas', (req, res) => {
17     return res.json(pessoas);
18 });
19
20
21 //http://localhost:3000
22 app.listen(3000);
```

Passo 2: testar se nosso código fonte funciona corretamente.

- Digitar primeiro **Ctrl + S** para salvar
- Depois digitar **Ctrl + C** na finalizar uma execução anterior do node, caso ela exista
- e depois digitar no terminal do VSCode o comando **node index.js**.

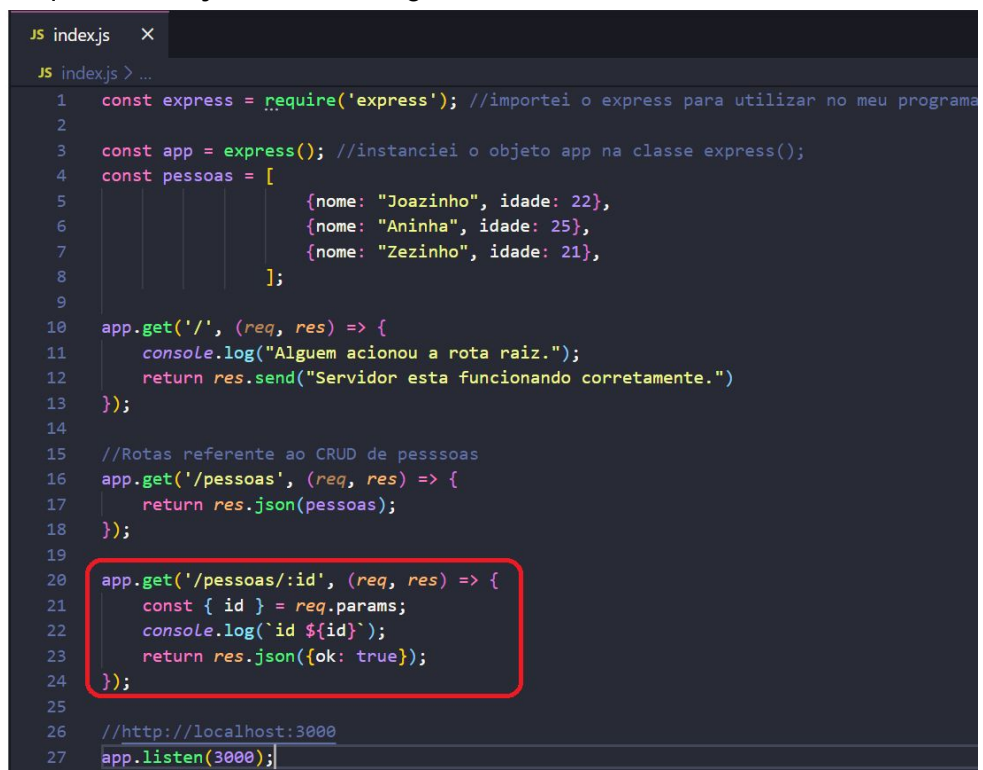
Passo 3: Abrir o navegador e digitar no endereço <http://localhost:3000/pessoas> . Se aparecer conforme a imagem abaixo, esta funcionando OK.


```
localhost:3000/pessoas  
[  
  - {  
    nome: "Joazinho",  
    idade: 22  
  },  
  - {  
    nome: "Aninha",  
    idade: 25  
  },  
  - {  
    nome: "Zezinho",  
    idade: 21  
  }  
]
```

Implementando a segunda Rota: GET localhost:3000/pessoas/id

Este endpoint é responsável por listar uma pessoa específica no serviço de pessoas.

Passo 1: Alterar o arquivo `index.js` conforme a figura abaixo.



```
JS index.js X  
JS index.js > ...  
1  const express = require('express'); //importei o express para utilizar no meu programa  
2  
3  const app = express(); //instanciei o objeto app na classe express()  
4  const pessoas = [  
5      {nome: "Joazinho", idade: 22},  
6      {nome: "Aninha", idade: 25},  
7      {nome: "Zezinho", idade: 21},  
8  ];  
9  
10 app.get('/', (req, res) => {  
11     console.log("Alguem acionou a rota raiz.");  
12     return res.send("Servidor esta funcionando corretamente.");  
13 });  
14  
15 //Rotas referente ao CRUD de pessoas  
16 app.get('/pessoas', (req, res) => {  
17     return res.json(pessoas);  
18 });  
19  
20 app.get('/pessoas/:id', (req, res) => {  
21     const { id } = req.params;  
22     console.log(`id ${id}`);  
23     return res.json({ok: true});  
24 });  
25  
26 //http://localhost:3000  
27 app.listen(3000);
```

Passo 2: testar se nosso código fonte funciona corretamente.

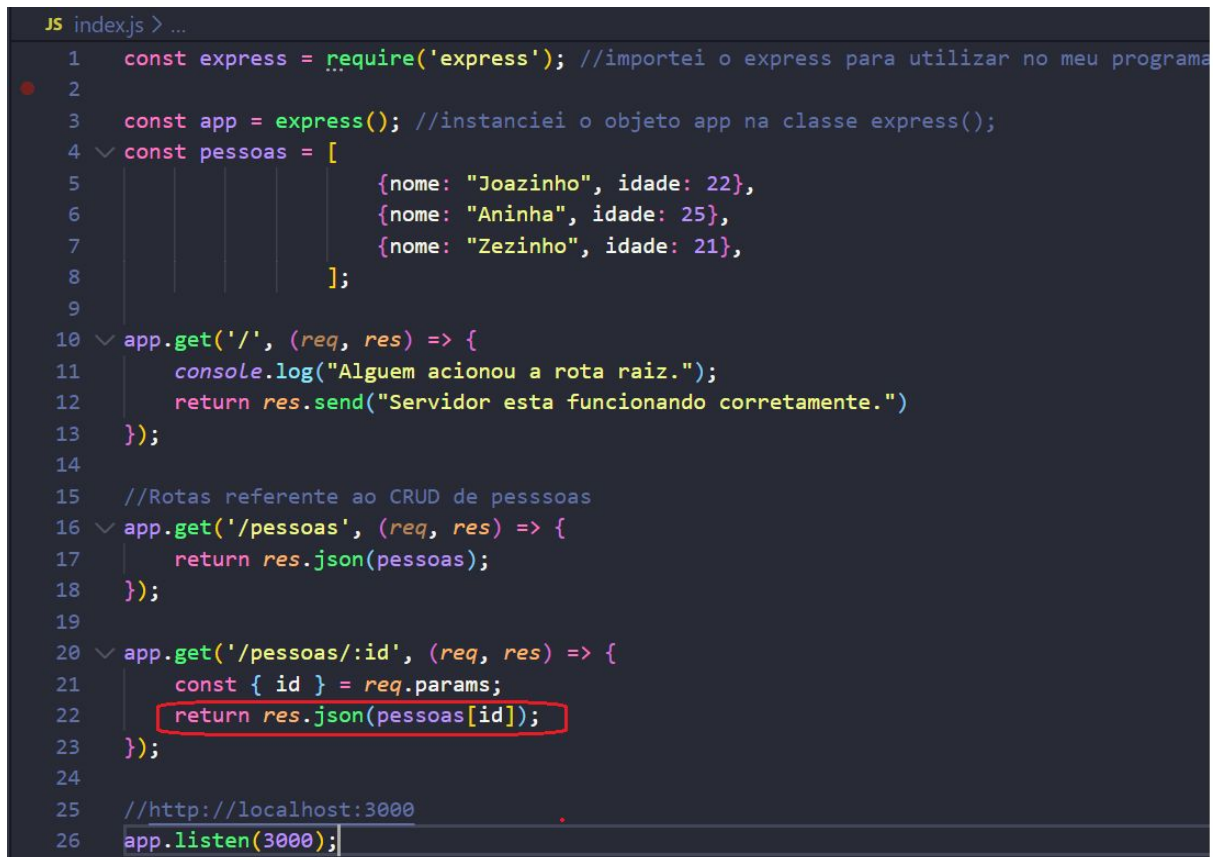
- Digitar primeiro **Ctrl + S** para salvar
- Depois digitar **Ctrl + C** na finalizar uma execução anterior do node, caso ela exista
- e depois digitar no terminal do VSCode o comando **node index.js**.

Passo 3: Abrir o navegador e digitar no endereço <http://localhost:3000/pessoas/1> .
Se aparecer conforme a imagem abaixo, esta funcionando OK.



```
{
  ok: true
}
```

Passo 4: Alterar o arquivo **index.js** conforme a figura abaixo.



```
JS index.js > ...
1  const express = require('express'); //importei o express para utilizar no meu programa
2
3  const app = express(); //instanciei o objeto app na classe express();
4  const pessoas = [
5      {nome: "Joazinho", idade: 22},
6      {nome: "Aninha", idade: 25},
7      {nome: "Zezinho", idade: 21},
8  ];
9
10 app.get('/', (req, res) => {
11     console.log("Alguem acionou a rota raiz.");
12     return res.send("Servidor esta funcionando corretamente.");
13 });
14
15 //Rotas referente ao CRUD de pessoas
16 app.get('/pessoas', (req, res) => {
17     return res.json(pessoas);
18 });
19
20 app.get('/pessoas/:id', (req, res) => {
21     const { id } = req.params;
22     return res.json(pessoas[id]);
23 });
24
25 //http://localhost:3000
26 app.listen(3000);
```

Passo 5: testar se nosso código fonte funciona corretamente.

- Digitar primeiro **Ctrl + S** para salvar
- Depois digitar **Ctrl + C** na finalizar uma execução anterior do node, caso ela exista
- e depois digitar no terminal do VSCode o comando **node index.js**.

Passo 3: Abrir o navegador e digitar no endereço <http://localhost:3000/pessoas/1> .
Se aparecer conforme a imagem abaixo, esta funcionando OK.



```
{
  nome: "Aninha",
  idade: 25
}
```