



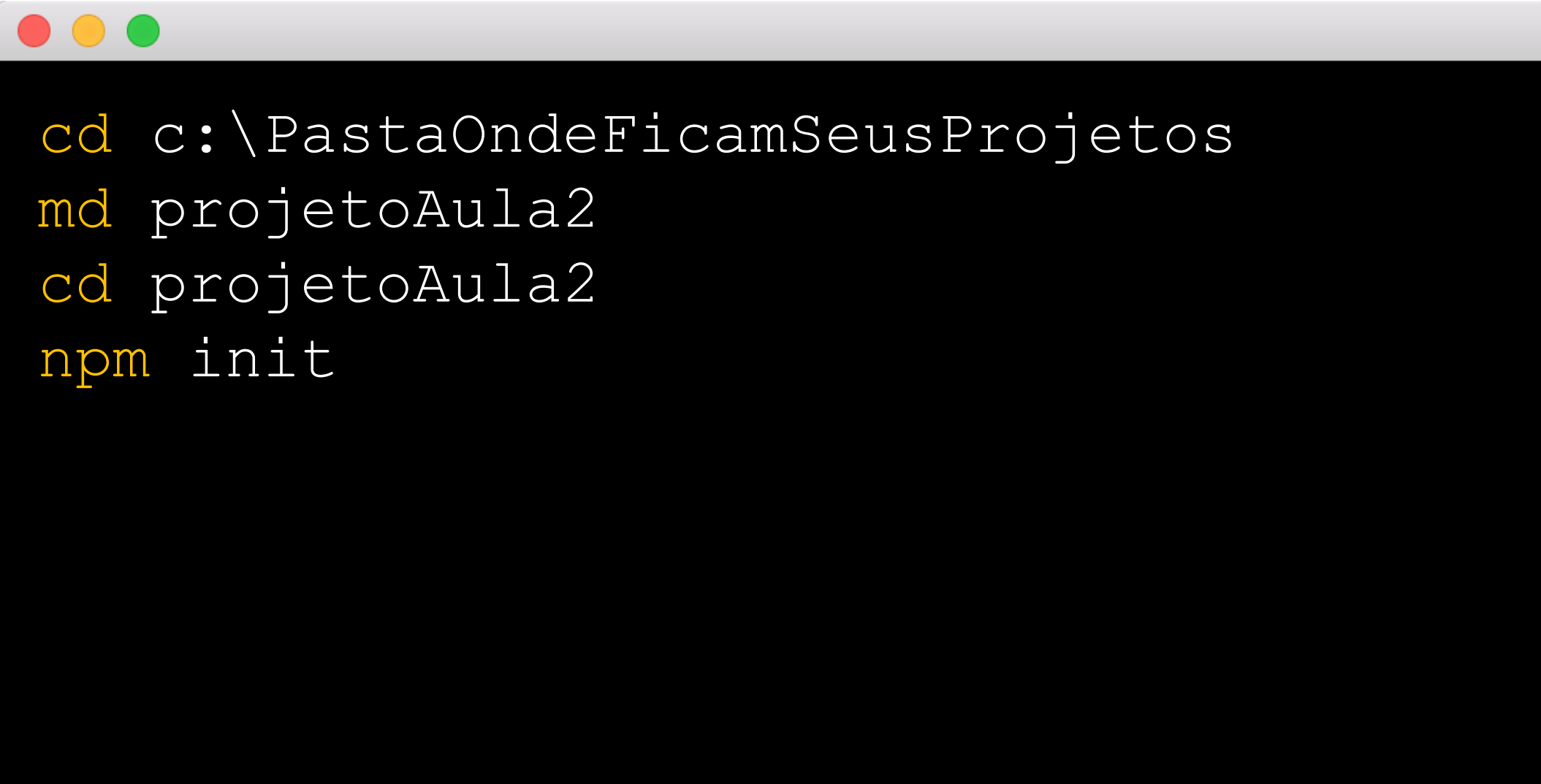
# Node.js

Sistemas de Informação  
Tópicos Avançados de Programação

2019.1

Bruno Catão

# Criando um novo projeto



```
cd c:\PastaOndeFicamSeusProjetos  
md projetoAula2  
cd projetoAula2  
npm init
```

# Criando um novo projeto



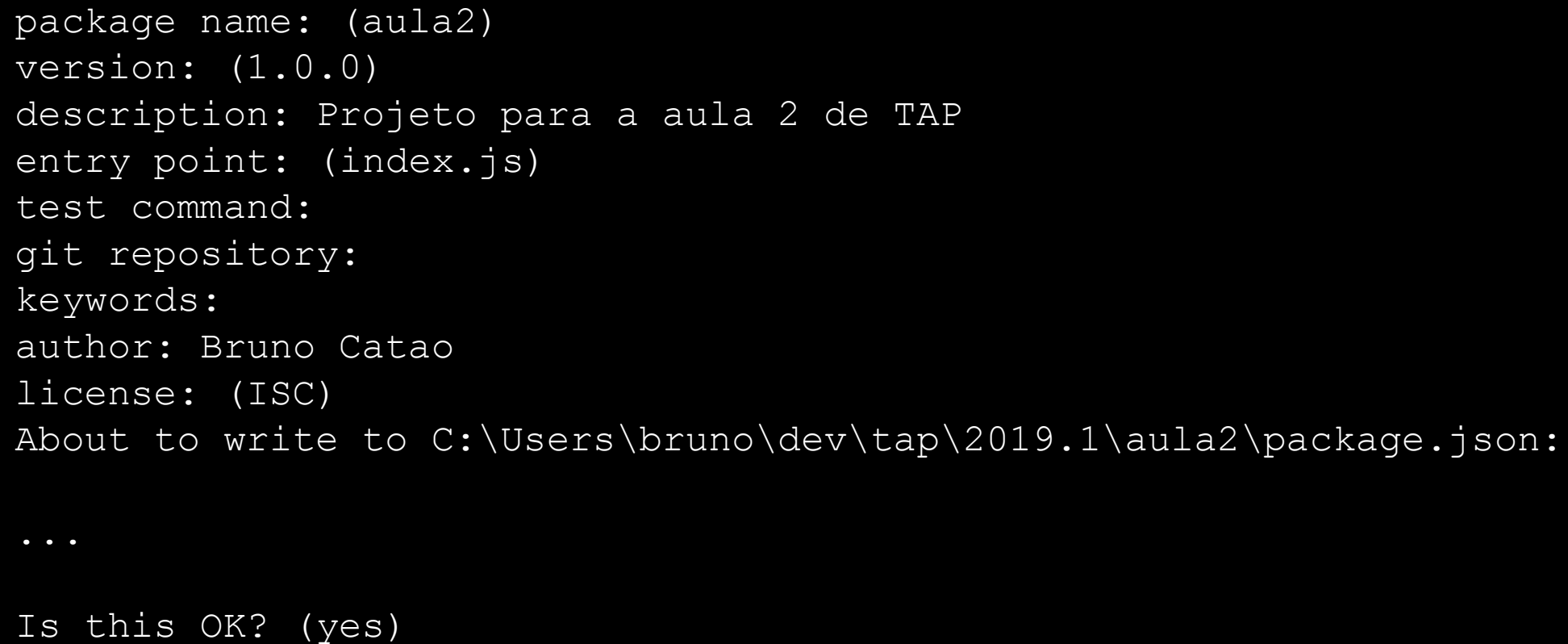
This utility will walk you through creating a package.json file. It only covers the most common items, and tries to guess sensible defaults.

See ``npm help json`` for definitive documentation on these fields and exactly what they do.

Use ``npm install <pkg>`` afterwards to install a package and save it as a dependency in the package.json file.

Press `^C` at any time to quit.

# Criando um novo projeto

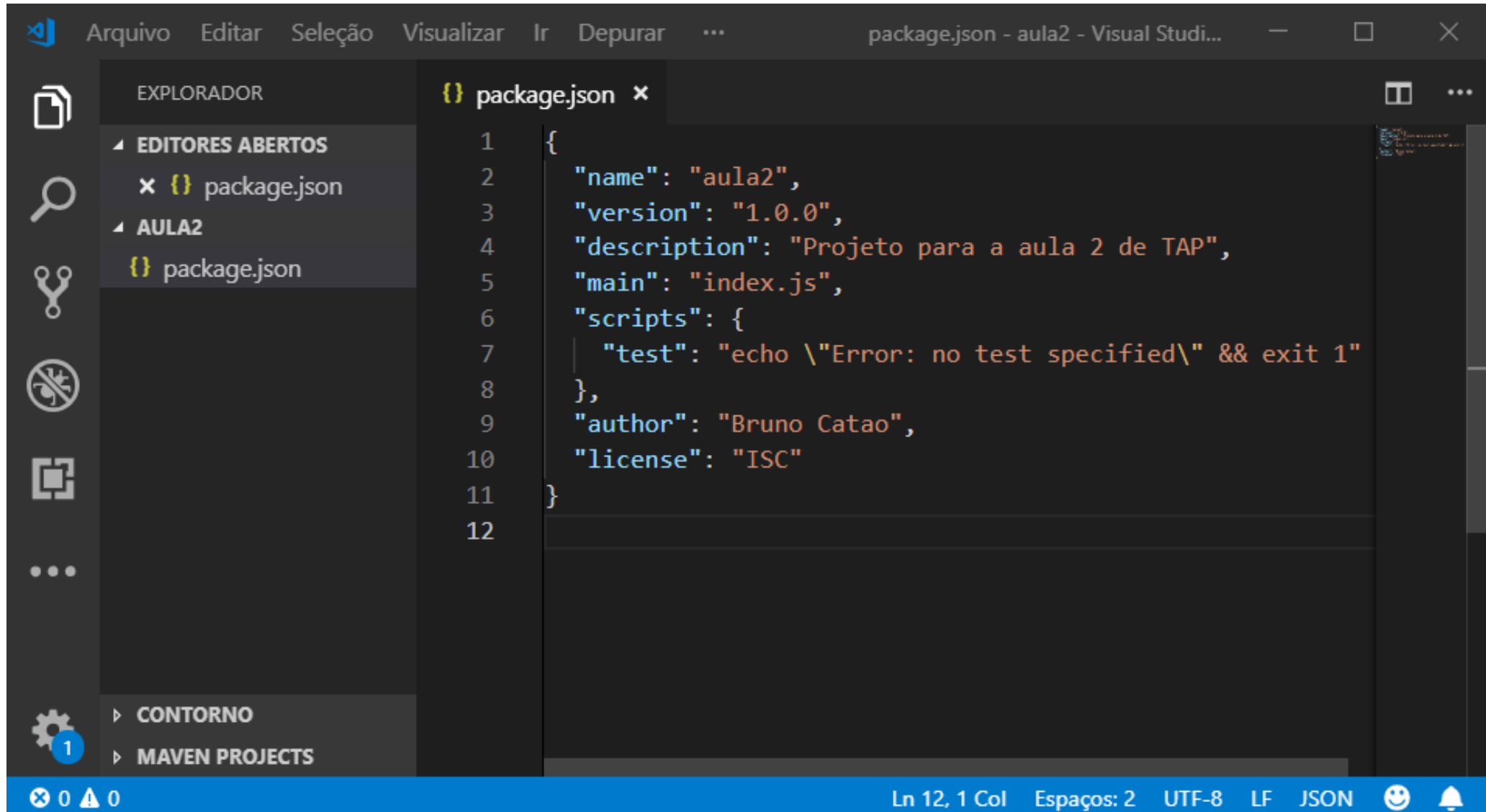


```
package name: (aula2)
version: (1.0.0)
description: Projeto para a aula 2 de TAP
entry point: (index.js)
test command:
git repository:
keywords:
author: Bruno Catao
license: (ISC)
About to write to C:\Users\bruno\dev\tap\2019.1\aula2\package.json:

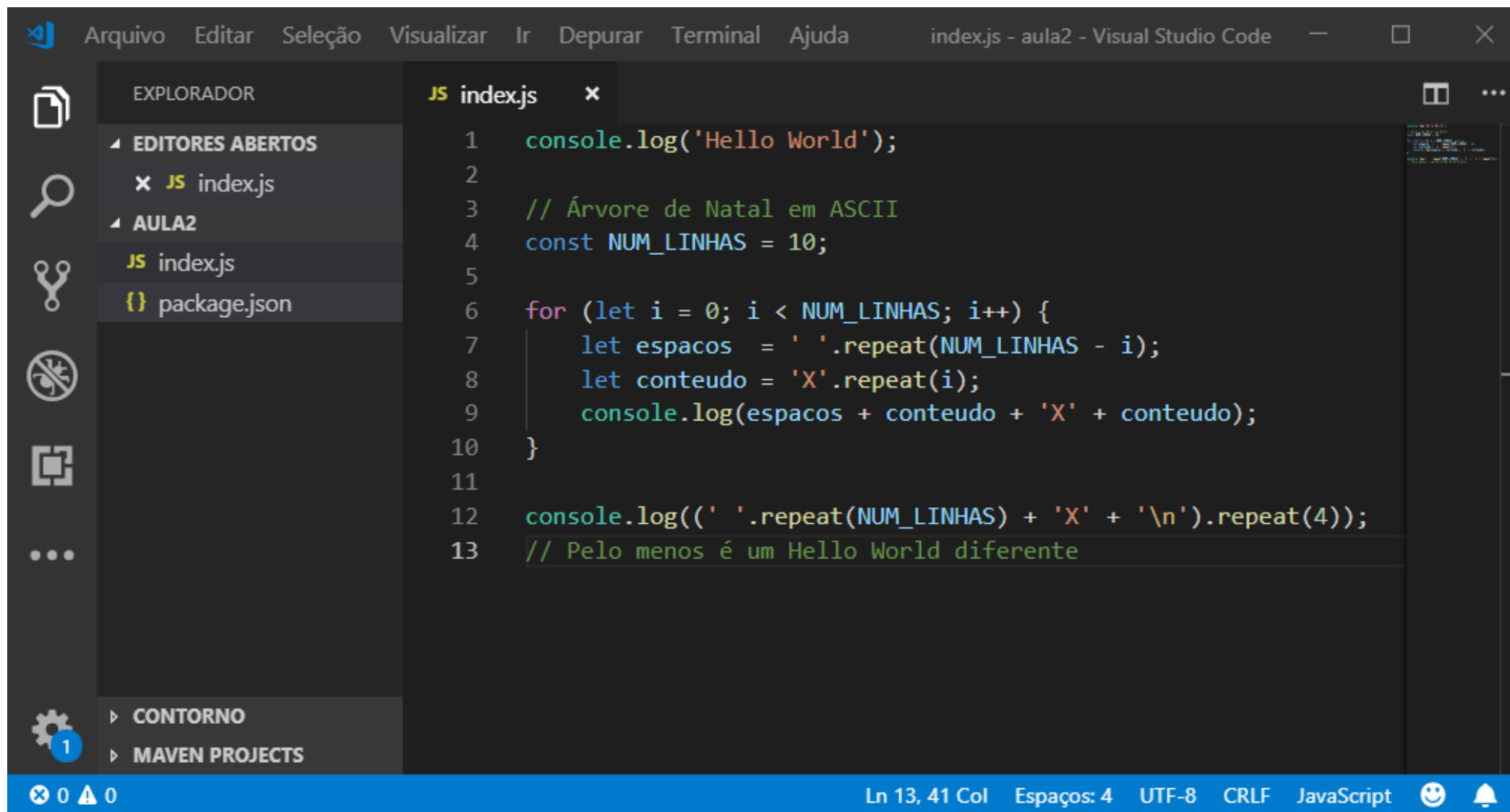
...

Is this OK? (yes)
```

# Estrutura do projeto



# Hello World



The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left shows a project named 'AULA2' containing 'index.js' and 'package.json'. The main editor area displays the code in 'index.js'.

```
1 console.log('Hello World');
2
3 // Árvore de Natal em ASCII
4 const NUM_LINHAS = 10;
5
6 for (let i = 0; i < NUM_LINHAS; i++) {
7     let espacos = ' '.repeat(NUM_LINHAS - i);
8     let conteudo = 'X'.repeat(i);
9     console.log(espacos + conteudo + 'X' + conteudo);
10 }
11
12 console.log((' '.repeat(NUM_LINHAS) + 'X' + '\n').repeat(4));
13 // Pelo menos é um Hello World diferente
```

The status bar at the bottom indicates the current position is Line 13, Column 41, with 4 spaces, UTF-8 encoding, CRLF line endings, and the JavaScript language mode.

# Executando

```
node index.js
```

```
Hello World
```

```
    X
```

```
   XXX
```

```
  XXXXX
```

```
 XXXXXXX
```

```
XXXXXXXXX
```

```
XXXXXXXXXX
```

```
XXXXXXXXXXXX
```

```
XXXXXXXXXXXXXX
```

```
XXXXXXXXXXXXXXXX
```

```
XXXXXXXXXXXXXXXXXX
```

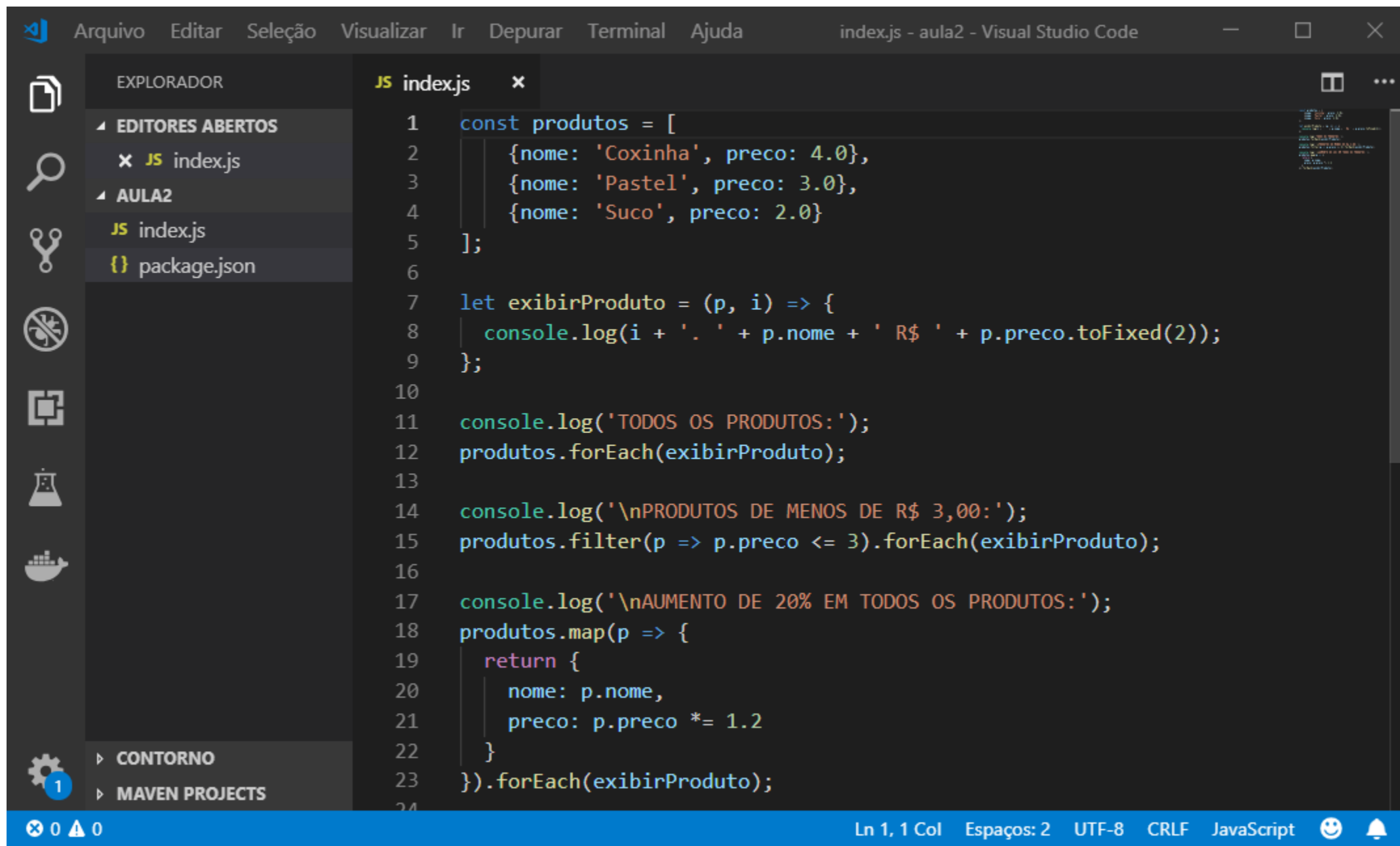
```
    X
```

```
    X
```

```
    X
```

```
    X
```

# JSON + Programação Funcional



```
1  const produtos = [
2      {nome: 'Coxinha', preco: 4.0},
3      {nome: 'Pastel', preco: 3.0},
4      {nome: 'Suco', preco: 2.0}
5  ];
6
7  let exibirProduto = (p, i) => {
8      console.log(i + ' ' + p.nome + ' R$ ' + p.preco.toFixed(2));
9  };
10
11 console.log('TODOS OS PRODUTOS:');
12 produtos.forEach(exibirProduto);
13
14 console.log('\nPRODUTOS DE MENOS DE R$ 3,00:');
15 produtos.filter(p => p.preco <= 3).forEach(exibirProduto);
16
17 console.log('\nAUMENTO DE 20% EM TODOS OS PRODUTOS:');
18 produtos.map(p => {
19     return {
20         nome: p.nome,
21         preco: p.preco * 1.2
22     }
23 }).forEach(exibirProduto);
24
```



# Executando

```
node index.js
```

```
TODOS OS PRODUTOS:
```

```
0. Coxinha R$ 4.00  
1. Pastel R$ 3.00  
2. Suco R$ 2.00
```

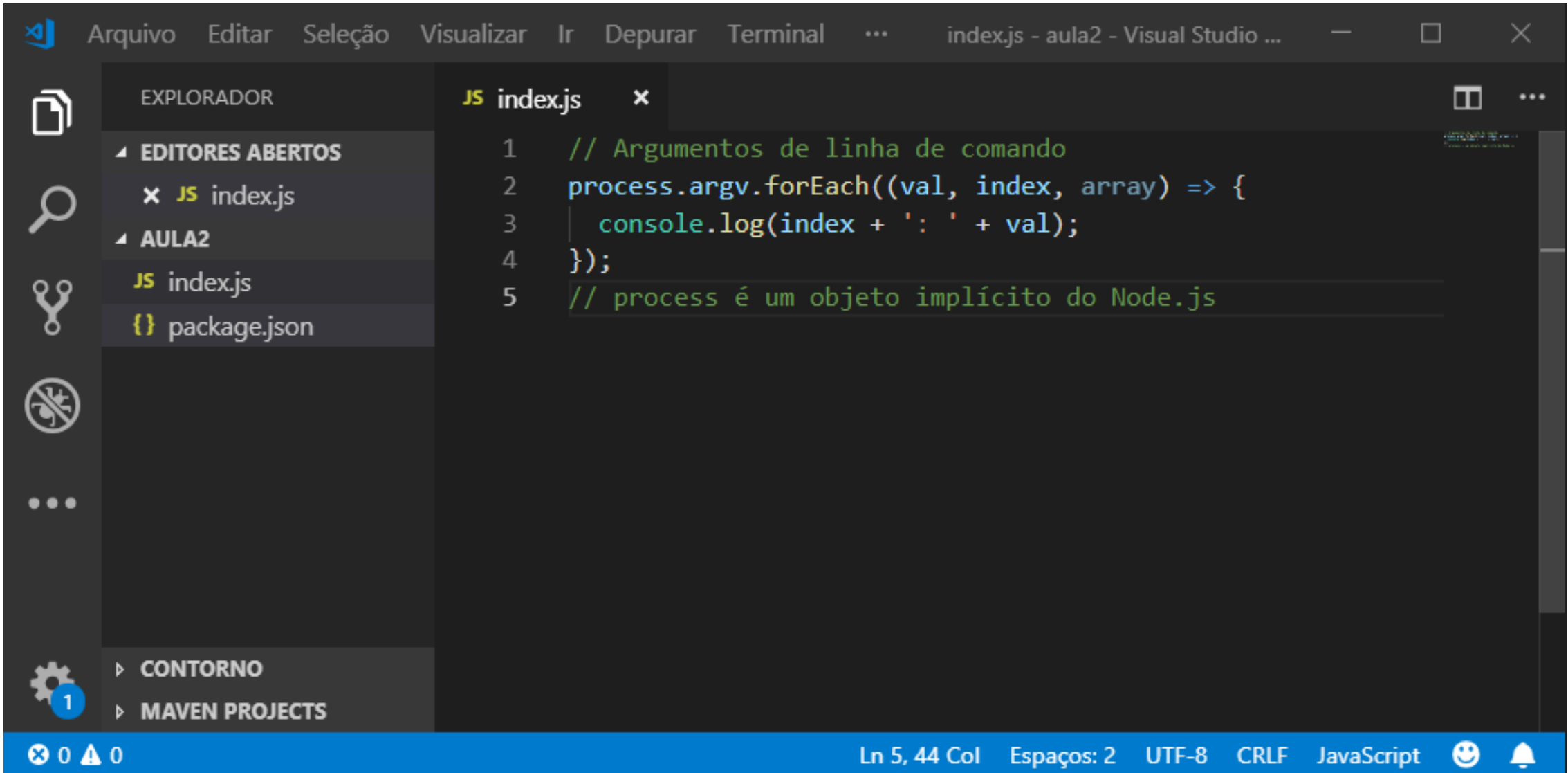
```
PRODUTOS DE MENOS DE R$ 3,00:
```

```
0. Pastel R$ 3.00  
1. Suco R$ 2.00
```

```
AUMENTO DE 20% EM TODOS OS PRODUTOS:
```

```
0. Coxinha R$ 4.80  
1. Pastel R$ 3.60  
2. Suco R$ 2.40
```

# Argumentos de linha de comando

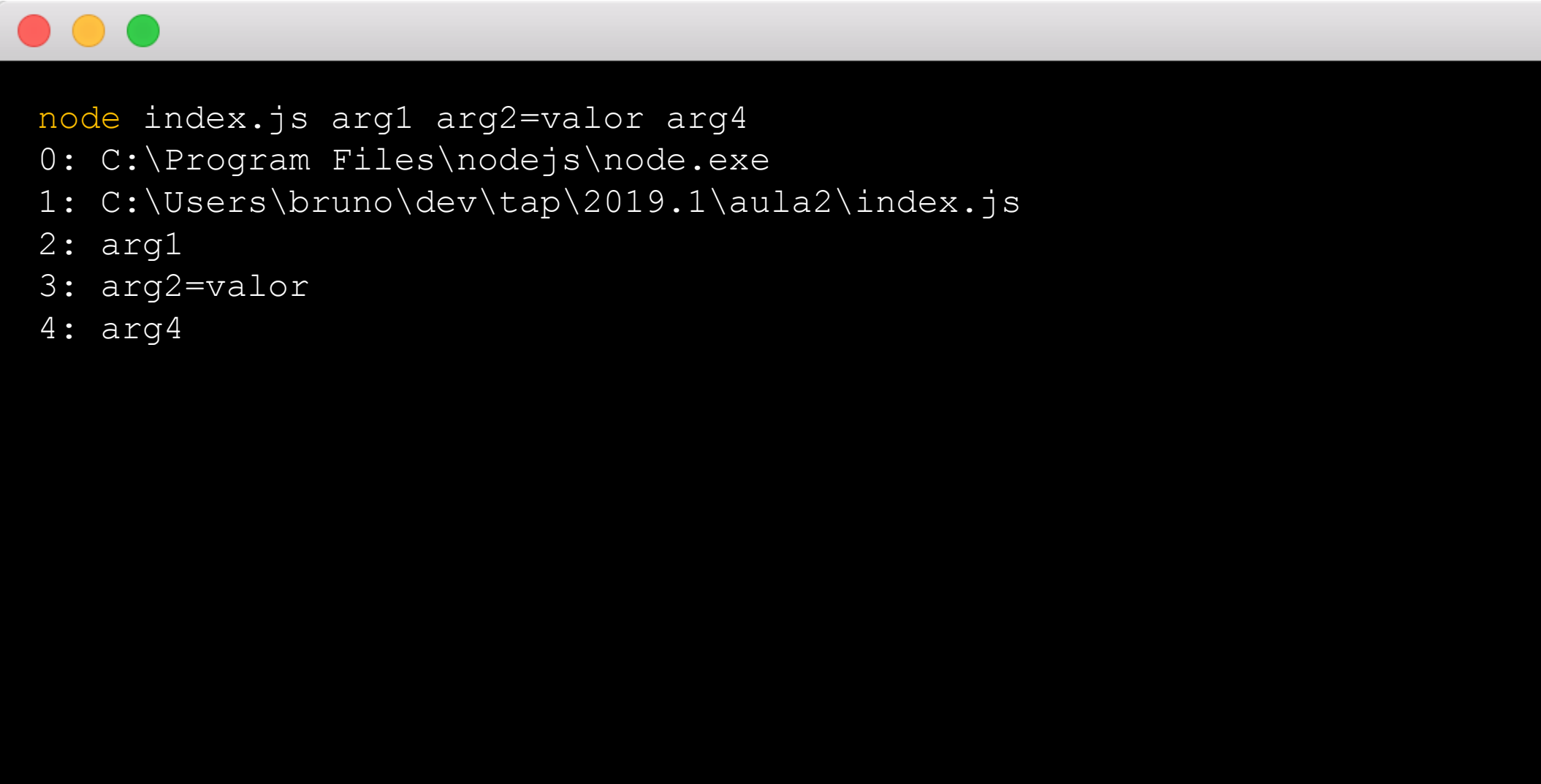


The image shows a screenshot of the Visual Studio Code editor interface. The top menu bar includes 'Arquivo', 'Editar', 'Seleção', 'Visualizar', 'Ir', 'Depurar', 'Terminal', and a dropdown menu. The title bar indicates the file is 'index.js - aula2 - Visual Studio ...'. The left sidebar contains the 'EXPLORADOR' (Explorer) view, which shows a project named 'AULA2' with two files: 'index.js' and 'package.json'. The 'EDITORES ABERTOS' (Open Editors) view shows 'index.js' is open. The main editor area displays the content of 'index.js', which is a JavaScript file. The code is as follows:

```
1 // Argumentos de linha de comando
2 process.argv.forEach((val, index, array) => {
3   console.log(index + ': ' + val);
4 });
5 // process é um objeto implícito do Node.js
```

The status bar at the bottom shows 'Ln 5, 44 Col', 'Espaços: 2', 'UTF-8', 'CRLF', 'JavaScript', and a smiley face icon.

# Executando

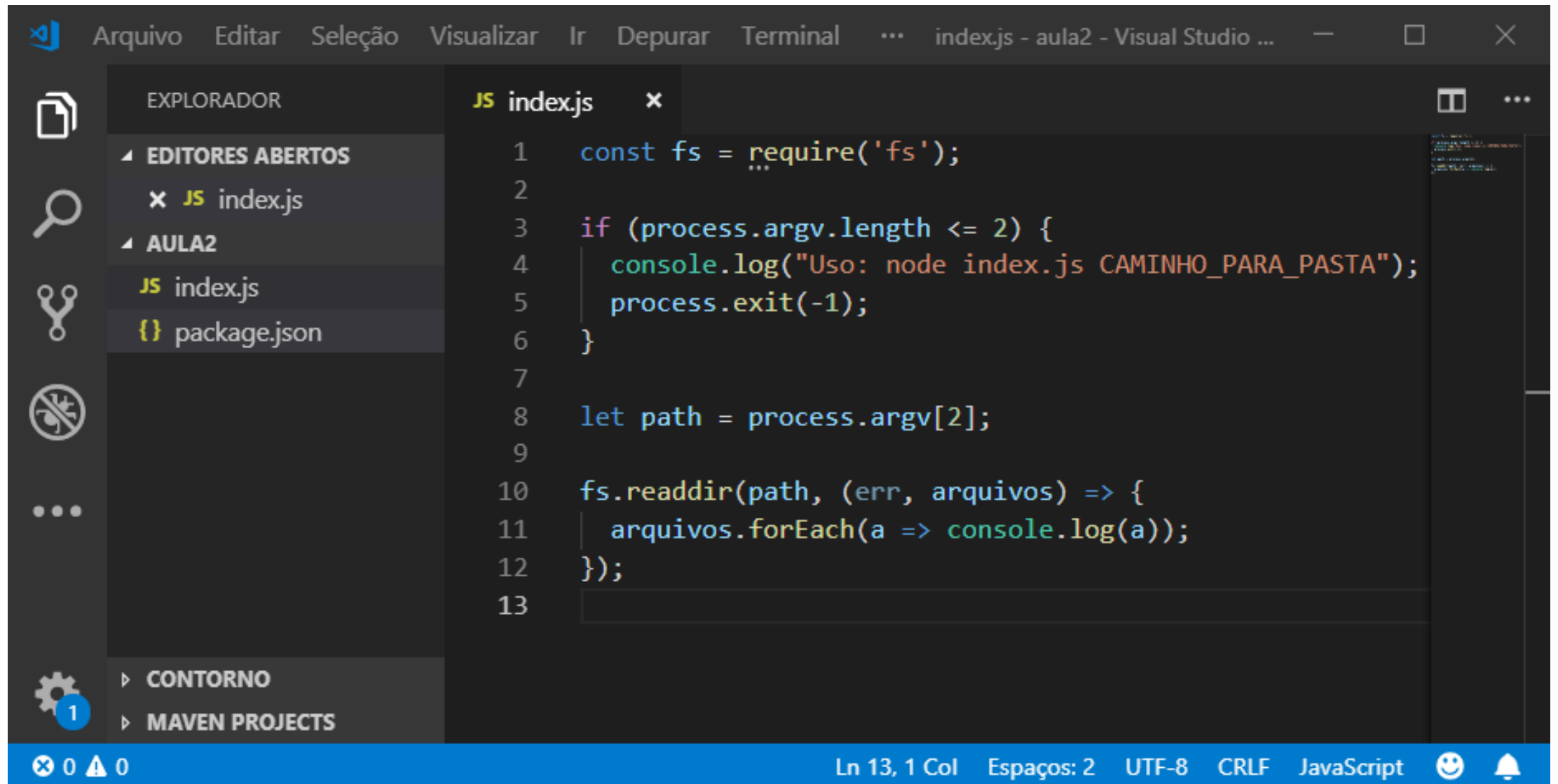


```
node index.js arg1 arg2=valor arg4  
0: C:\Program Files\nodejs\node.exe  
1: C:\Users\bruno\dev\tap\2019.1\aula2\index.js  
2: arg1  
3: arg2=valor  
4: arg4
```

# Objeto Process (<https://nodejs.org/api/process.html>)

- Eventos:
  - **beforeExit** – Executar uma função antes de encerrar a execução
- Funções:
  - **abort()** – Força a interrupção da execução imediatamente
  - **exit(valor)** – Envia um sinal para o SO interromper o processo
  - **chdir()** – Muda o diretório atual
- Atributos:
  - **arch** – Exibe atributos da arquitetura de hardware
  - **argv** – Objeto com os argumentos da linha de comando
  - **env** – Objeto com as variáveis de ambiente

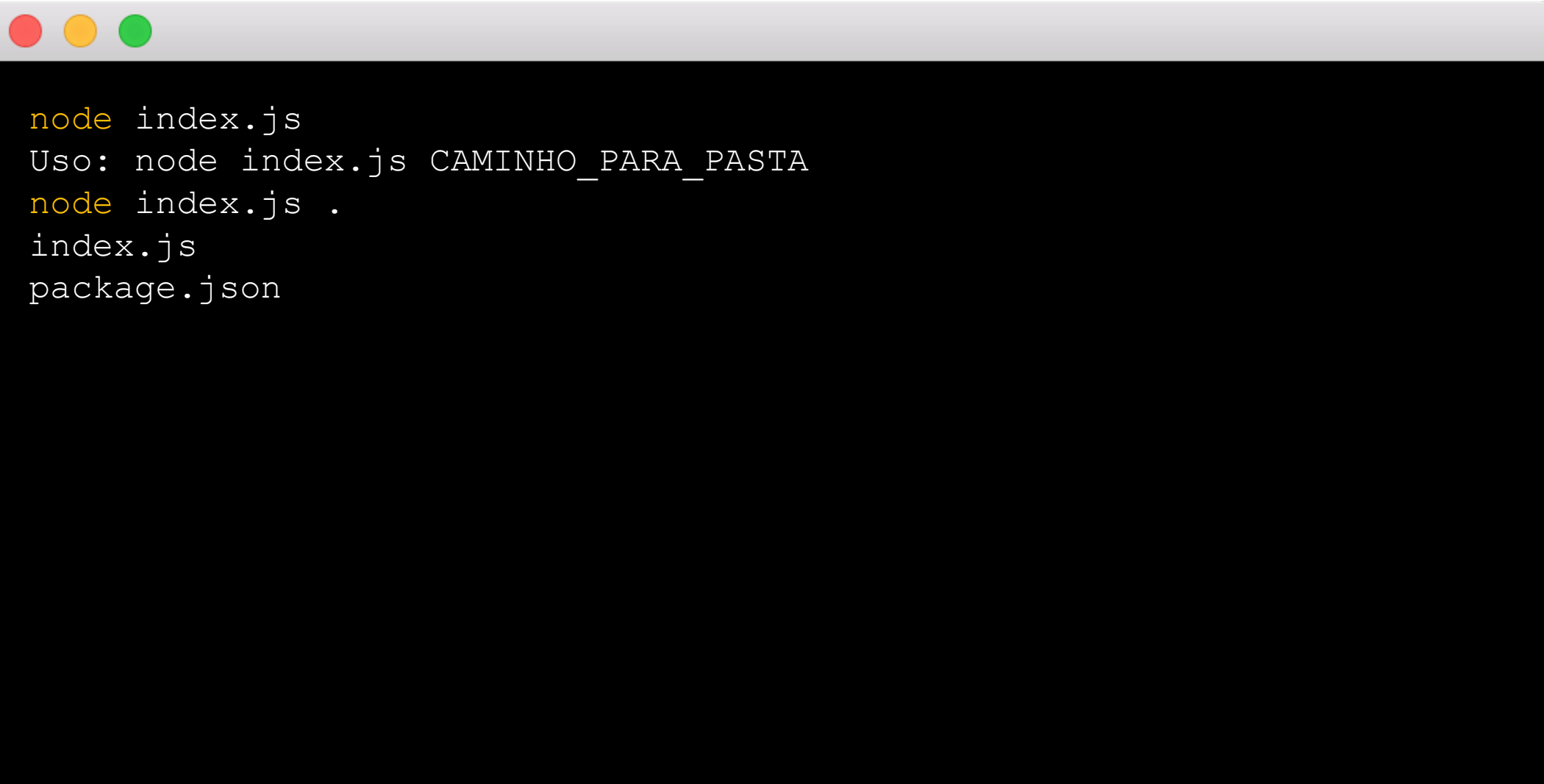
# Listando arquivos de uma pasta



The screenshot shows the Visual Studio Code interface. On the left, the 'EXPLORADOR' (Explorer) sidebar is open, displaying the file structure of a project named 'AULA2'. It shows two files: 'index.js' and 'package.json'. The 'index.js' file is selected. The main editor area displays the contents of 'index.js', which is a JavaScript file. The code uses the 'fs' module to read the contents of a directory. The status bar at the bottom indicates the current line and column (Ln 13, 1 Col), the encoding (UTF-8), the line endings (CRLF), and the language (JavaScript).

```
1  const fs = require('fs');
2
3  if (process.argv.length <= 2) {
4      console.log("Uso: node index.js CAMINHO_PARA_PASTA");
5      process.exit(-1);
6  }
7
8  let path = process.argv[2];
9
10 fs.readdir(path, (err, arquivos) => {
11     arquivos.forEach(a => console.log(a));
12 });
13
```

# Executando

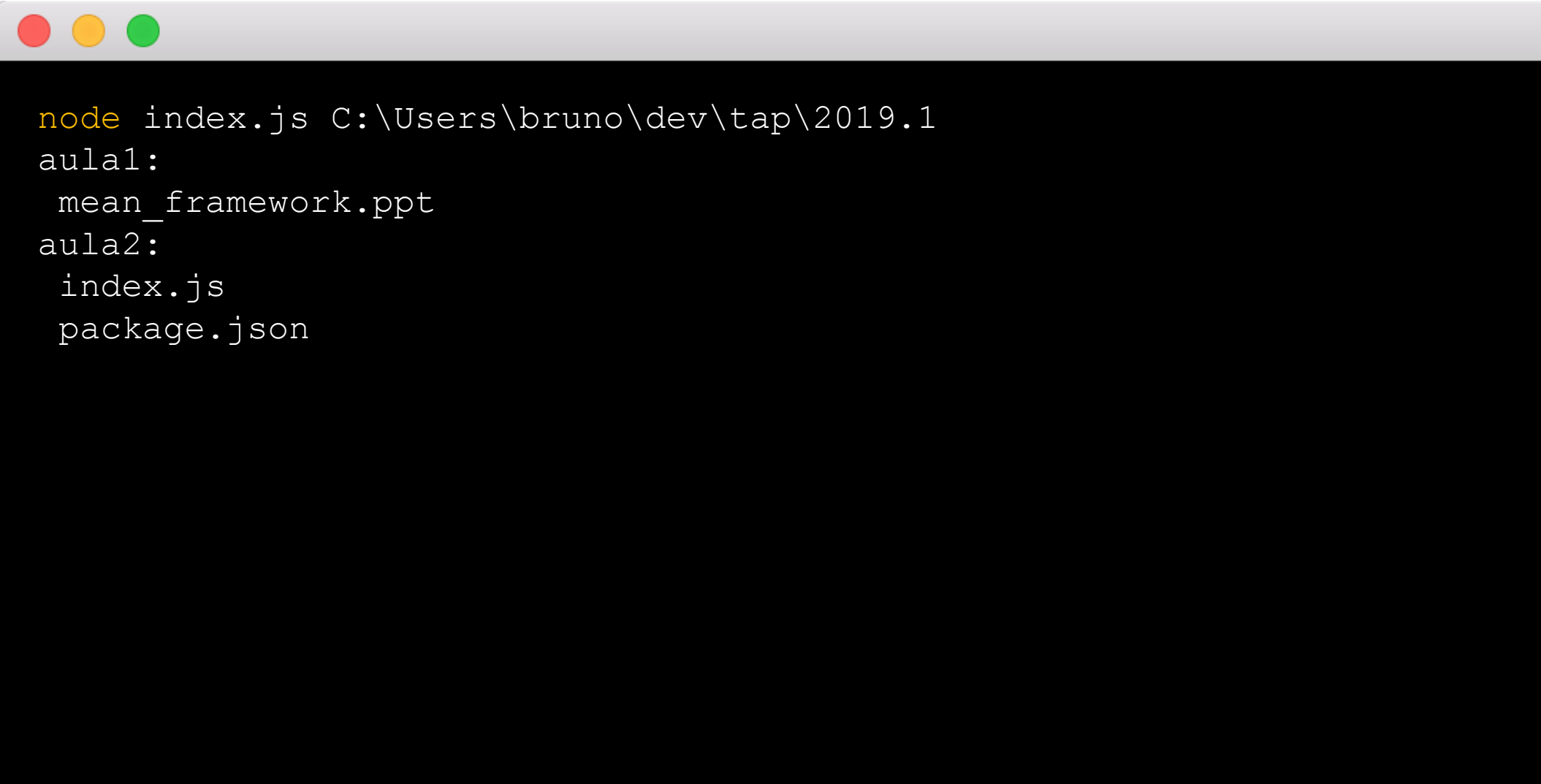
A terminal window with a light gray title bar containing three colored window control buttons (red, yellow, green). The terminal background is black, and the text is white. The text inside the terminal shows a command to run a JavaScript file, its usage, and the files it depends on.

```
node index.js
Uso: node index.js CAMINHO_PARA_PASTA
node index.js .
index.js
package.json
```

# Versão recursiva – percorre subpastas

```
7 // Versão recursiva - lista subpastas
8 let pasta = process.argv[2];
9
10 let listarArquivos = (pasta, arquivos, nivel) => {
11   arquivos.forEach(a => {
12     if (fs.statSync(pasta + '/' + a).isDirectory()) {
13       console.log(' '.repeat(nivel) + a + ':');
14       let subPasta = pasta + '/' + a;
15       fs.readdir(subPasta, (err, arquivosSubPasta) => {
16         listarArquivos(subPasta, arquivosSubPasta, nivel + 1);
17       });
18     } else {
19       console.log(' '.repeat(nivel) + a);
20     }
21   });
22 };
23
24 fs.readdir(pasta, (err, arquivos) => {
25   listarArquivos(pasta, arquivos, 0);
26 });
```

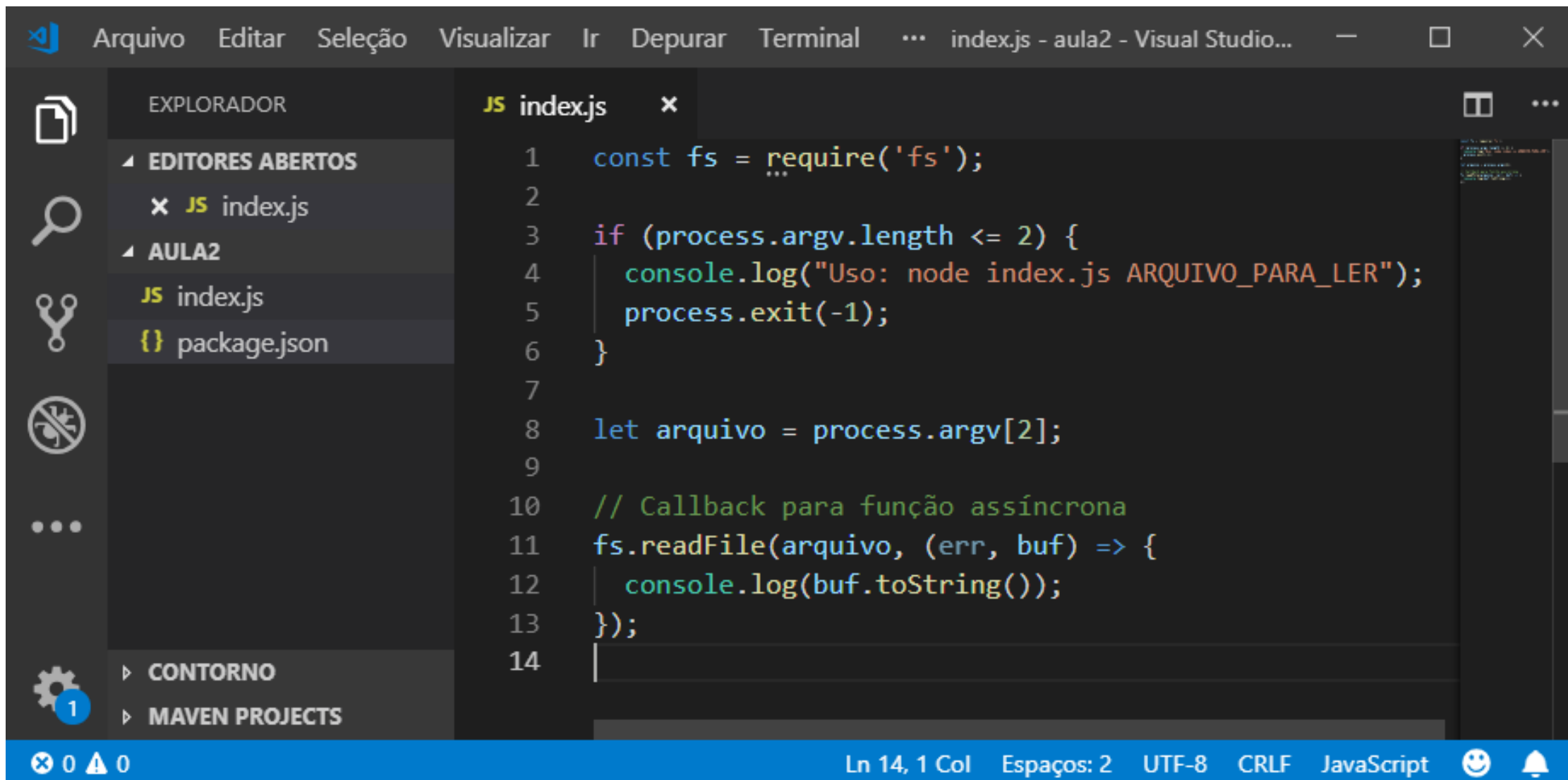
# Executando



```
node index.js C:\Users\bruno\dev\tap\2019.1  
aula1:  
  mean_framework.ppt  
aula2:  
  index.js  
  package.json
```



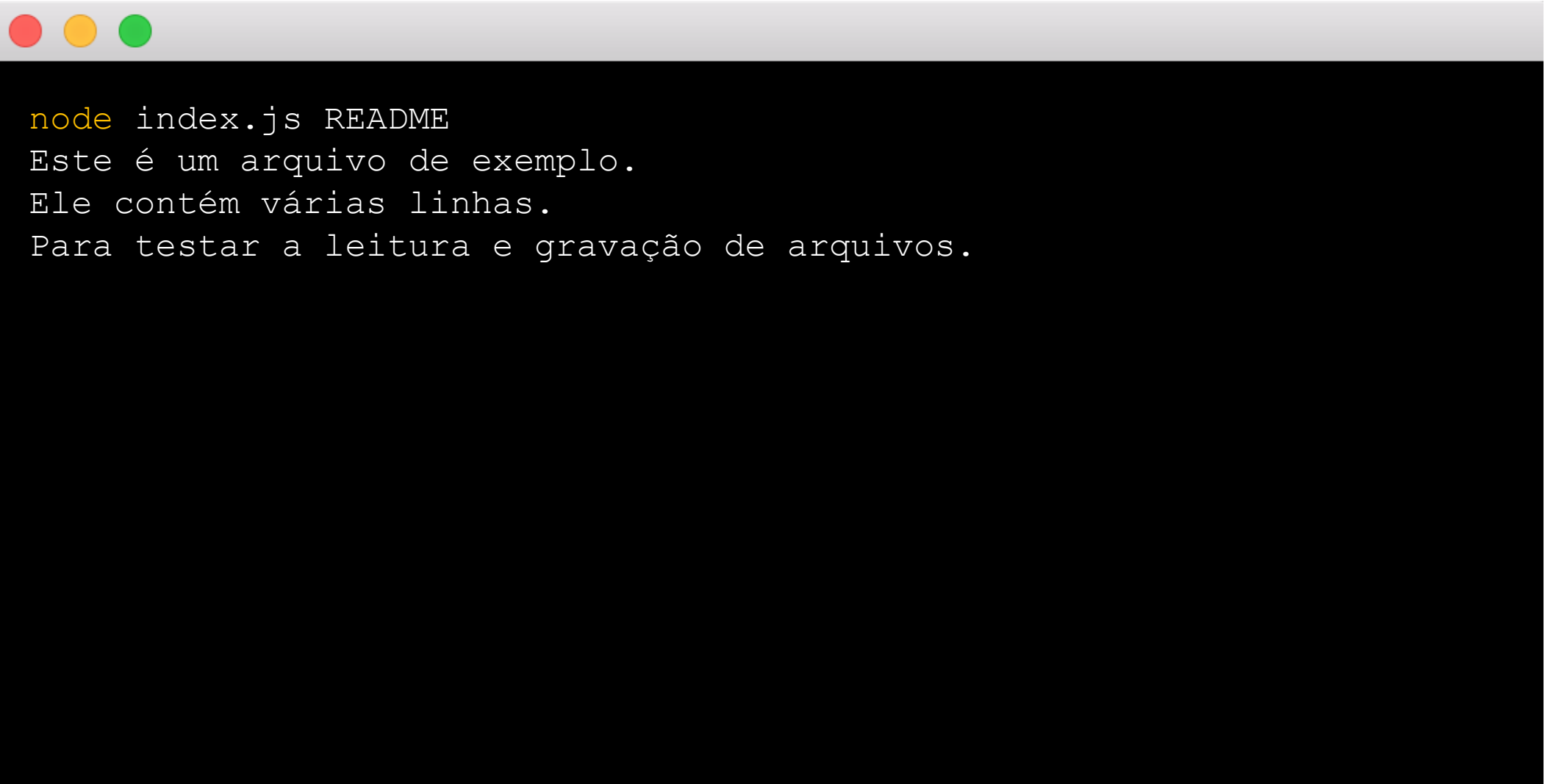
# Lendo um arquivo – Funções assíncronas



```
Arquivo  Editar  Seleção  Visualizar  Ir  Depurar  Terminal  ...  index.js - aula2 - Visual Studio...  
  
EXPLORADOR  
└─ EDITORES ABERTOS  
    └─ JS index.js  
└─ AULA2  
    └─ JS index.js  
    └─ {} package.json  
  
└─ CONTORNO  
└─ MAVEN PROJECTS  
  
1  const fs = require('fs');  
2  
3  if (process.argv.length <= 2) {  
4      console.log("Uso: node index.js ARQUIVO_PARA_LER");  
5      process.exit(-1);  
6  }  
7  
8  let arquivo = process.argv[2];  
9  
10 // Callback para função assíncrona  
11 fs.readFile(arquivo, (err, buf) => {  
12     console.log(buf.toString());  
13 });  
14
```

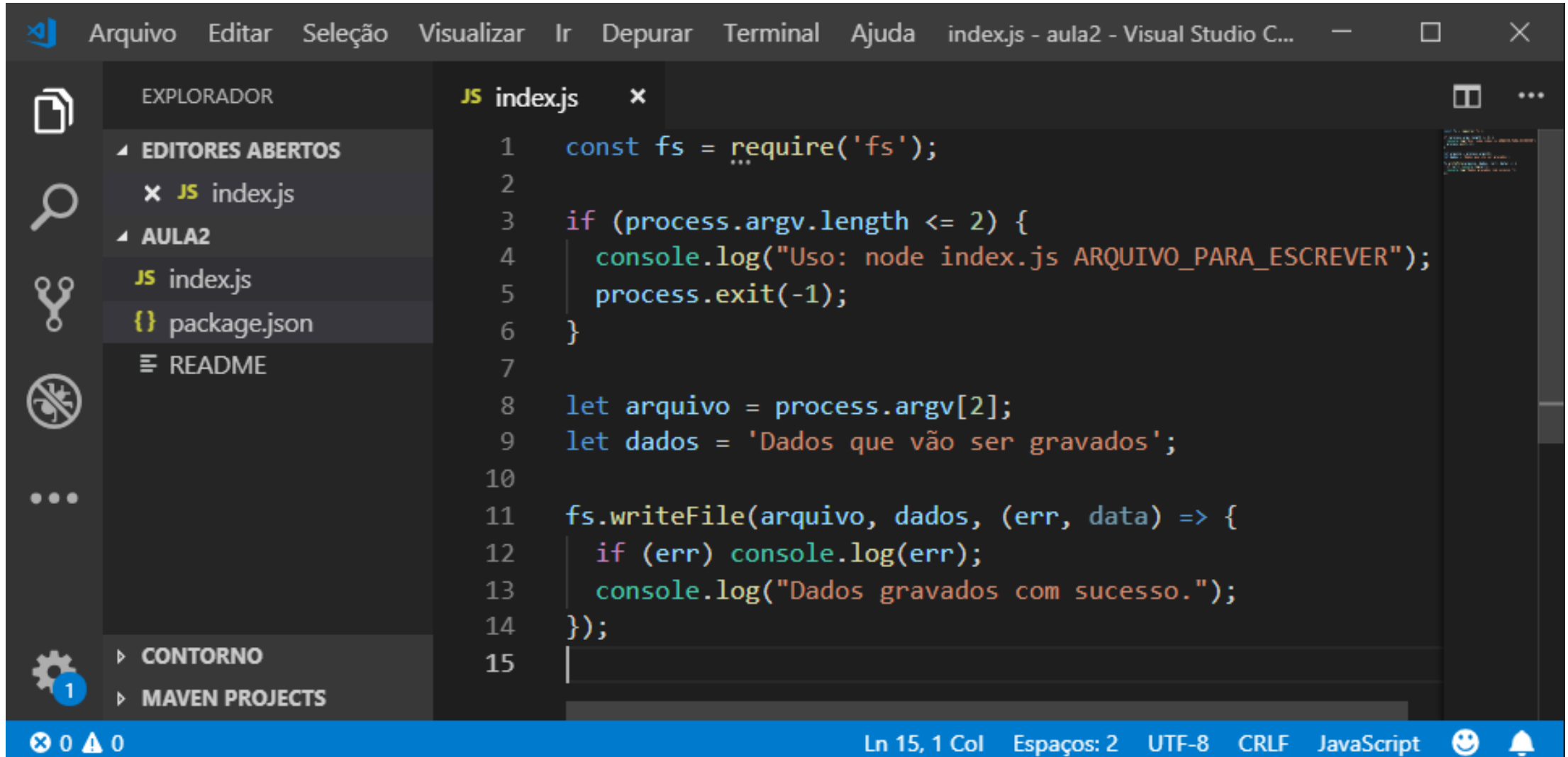
Ln 14, 1 Col Espaços: 2 UTF-8 CRLF JavaScript

# Executando



```
node index.js README
Este é um arquivo de exemplo.
Ele contém várias linhas.
Para testar a leitura e gravação de arquivos.
```

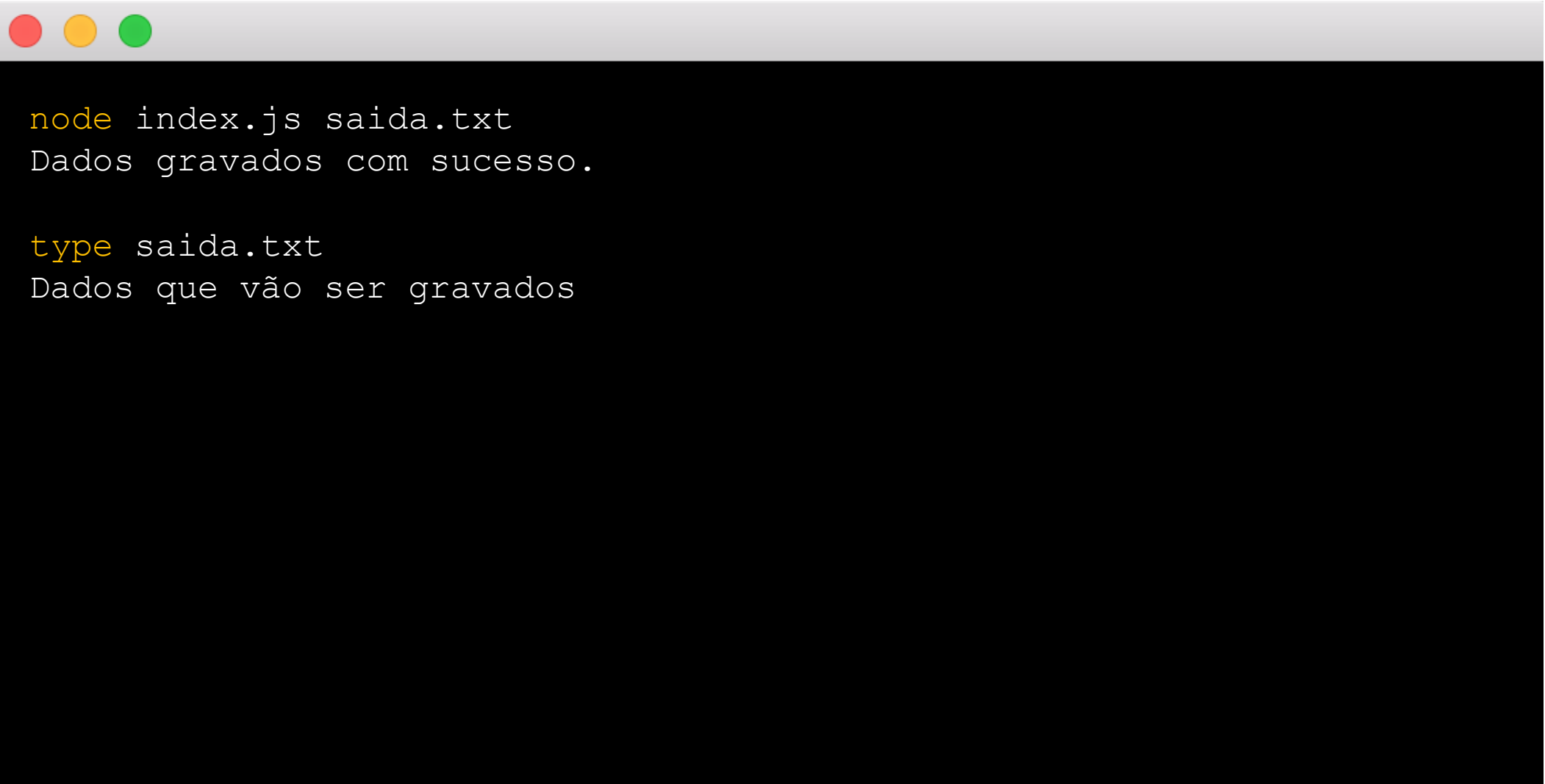
# Gravando um arquivo – Funções assíncronas



```
Arquivo  Editar  Seleção  Visualizar  Ir  Depurar  Terminal  Ajuda  index.js - aula2 - Visual Studio C...  
EXPLORADOR  
EDITORES ABERTOS  
x JS index.js  
AULA2  
JS index.js  
{ } package.json  
≡ README  
CONTORNO  
MAVEN PROJECTS  
1  
1  const fs = require('fs');  
2  
3  if (process.argv.length <= 2) {  
4      console.log("Uso: node index.js ARQUIVO_PARA_ESCREVER");  
5      process.exit(-1);  
6  }  
7  
8  let arquivo = process.argv[2];  
9  let dados = 'Dados que vão ser gravados';  
10  
11  fs.writeFile(arquivo, dados, (err, data) => {  
12      if (err) console.log(err);  
13      console.log("Dados gravados com sucesso.");  
14  });  
15
```

Ln 15, 1 Col Espaços: 2 UTF-8 CRLF JavaScript

# Executando

A terminal window with a light gray title bar containing three colored window control buttons (red, yellow, green). The terminal background is black, and the text is white. The text shows two commands being executed: 'node index.js saida.txt' followed by the output 'Dados gravados com sucesso.', and 'type saida.txt' followed by the output 'Dados que vão ser gravados'.

```
node index.js saida.txt
Dados gravados com sucesso.

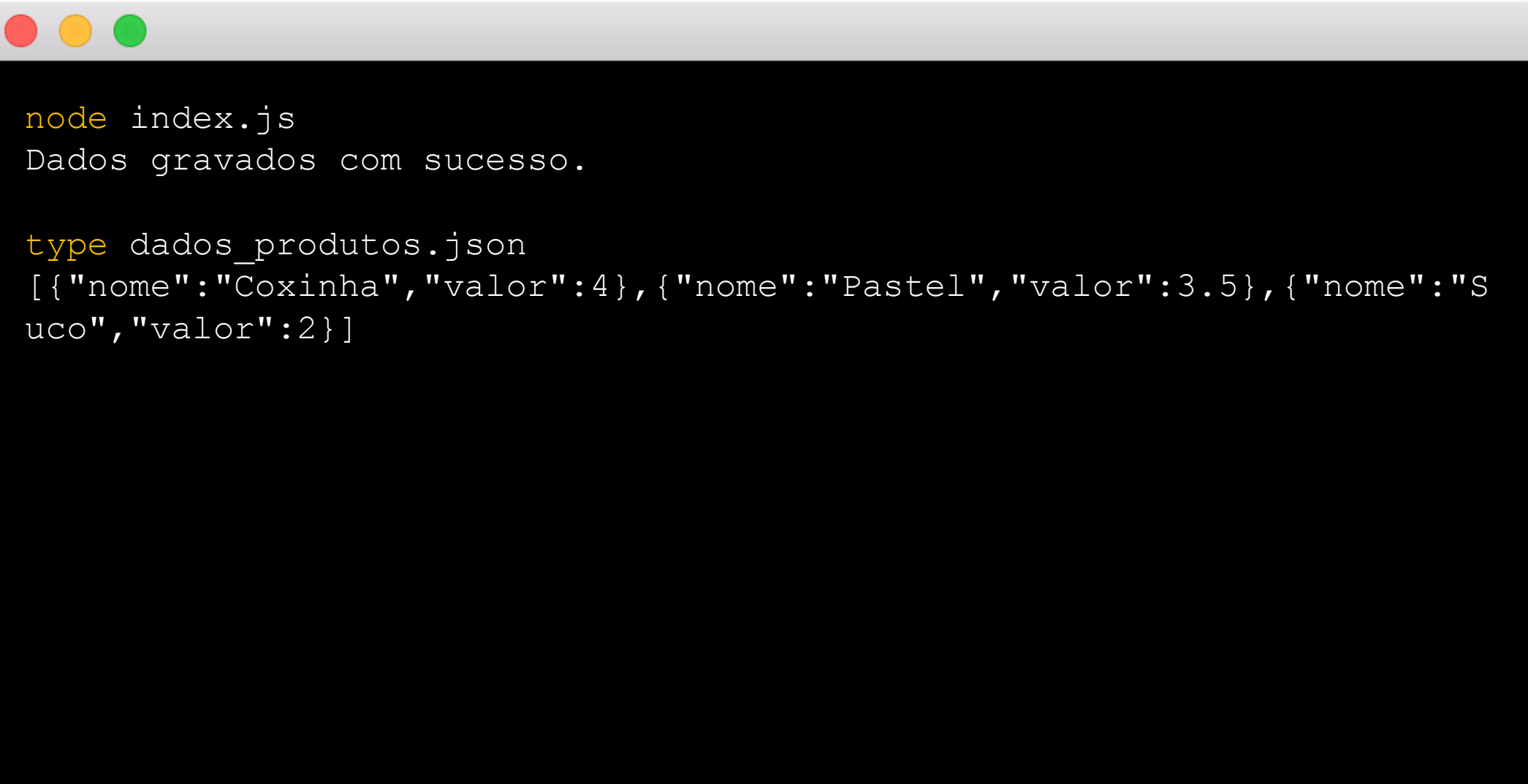
type saida.txt
Dados que vão ser gravados
```

# Exemplo Prático – Gravando objetos

JS index.js x

```
1  const fs = require('fs');
2
3  const ARQUIVO_BD = 'dados_produtos.json';
4
5  let produtos = [
6    {nome: 'Coxinha', valor: 4.0},
7    {nome: 'Pastel', valor: 3.5},
8    {nome: 'Suco', valor: 2.0}
9  ];
10
11  // Gravando o banco de dados
12  fs.writeFile(ARQUIVO_BD, JSON.stringify(produtos), (err, data) => {
13    if (err) console.log(err);
14    console.log("Dados gravados com sucesso.");
15  });
```

# Executando



```
node index.js
```

```
Dados gravados com sucesso.
```

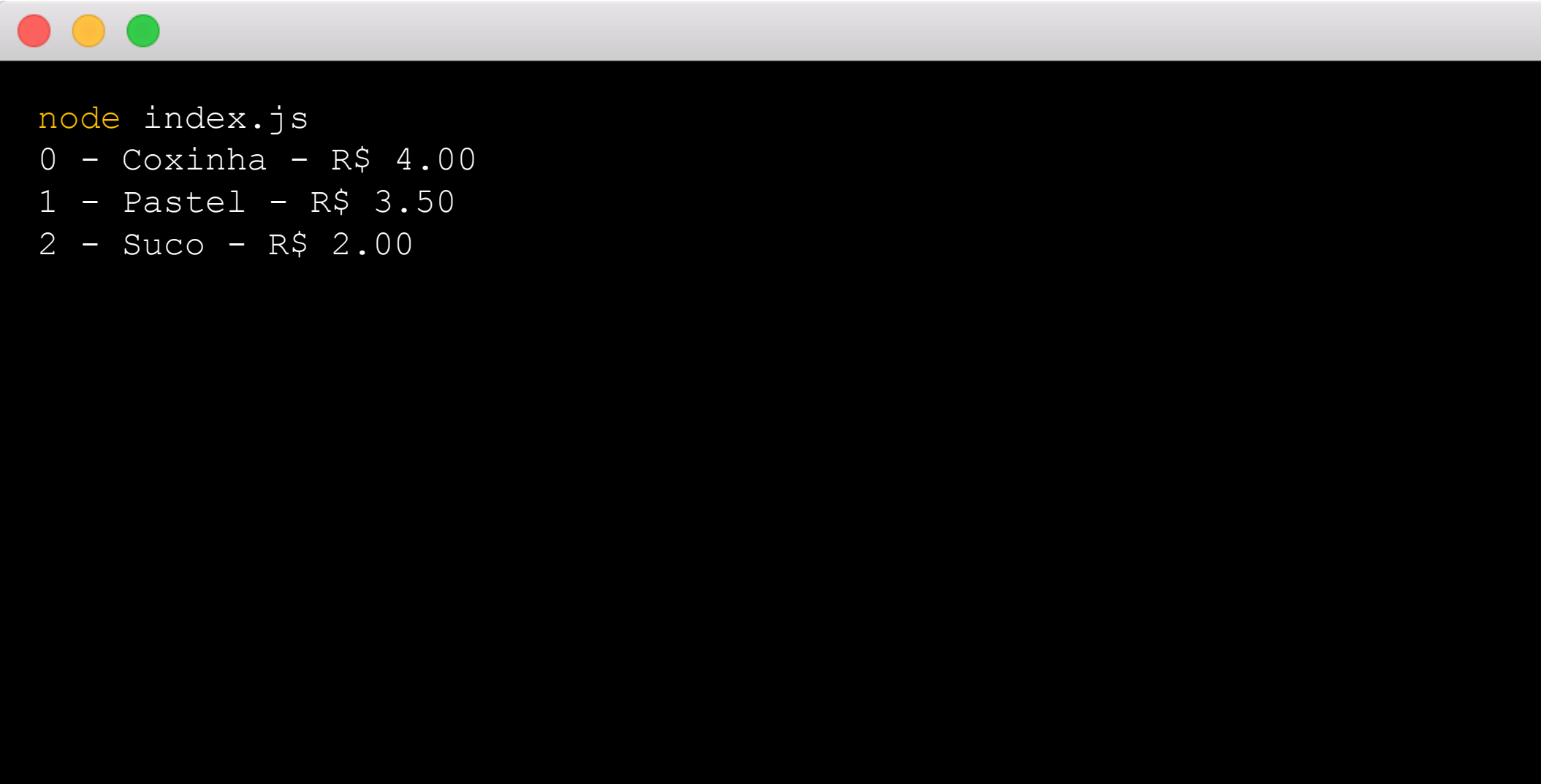
```
type dados_produtos.json
```

```
[{"nome": "Coxinha", "valor": 4}, {"nome": "Pastel", "valor": 3.5}, {"nome": "S  
uco", "valor": 2}]
```

# Exemplo Prático – Lendo objetos

```
JS index.js  ×
1  const fs = require('fs');
2
3  const ARQUIVO_BD = 'dados_produtos.json';
4
5  // Lendo o arquivo do banco de dados
6  fs.readFile(ARQUIVO_BD, (err, buf) => {
7      if (err) {
8          console.log('Não foi possível ler os dados', err);
9      } else {
10         let produtos = JSON.parse(buf.toString());
11         produtos.forEach((p, i) => {
12             console.log(i + ' - ' + p.nome + ' - R$ ' + p.valor.toFixed(2));
13         });
14     }
15 });
```

# Executando

A terminal window with a light gray title bar containing three colored window control buttons (red, yellow, green). The terminal area has a black background with white text.

```
node index.js
```

```
0 - Coxinha - R$ 4.00
```

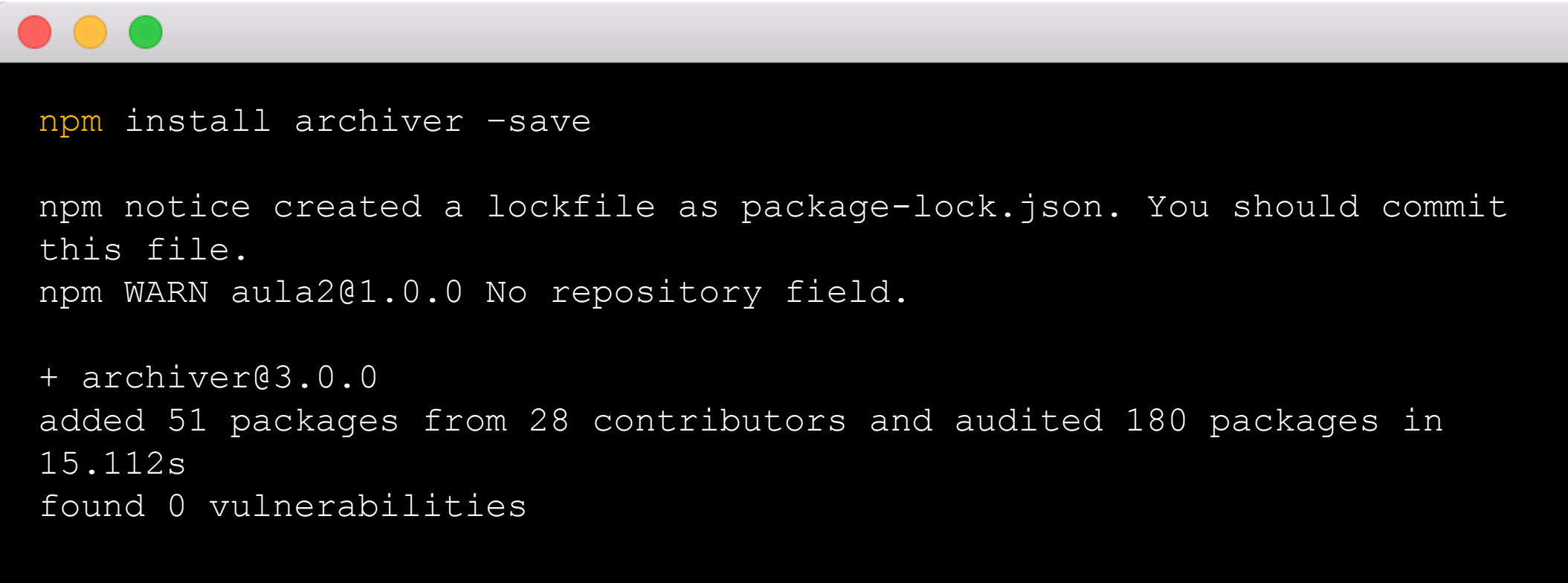
```
1 - Pastel - R$ 3.50
```

```
2 - Suco - R$ 2.00
```



# Exemplo Prático #2 – Compactando arquivos

- Vamos instalar um pacote no nosso projeto
- Para isso, utilizamos o comando “`npm install <NOME_DO_PACOTE>`”
- O argumento “`-save`” salva o pacote no arquivo `package.json`



```
npm install archiver -save
```

```
npm notice created a lockfile as package-lock.json. You should commit this file.
```

```
npm WARN aula2@1.0.0 No repository field.
```

```
+ archiver@3.0.0
```

```
added 51 packages from 28 contributors and audited 180 packages in 15.112s
```

```
found 0 vulnerabilities
```

# Onde os pacotes são instalados ?

- Pacotes locais:
  - Pasta `node_modules` do seu projeto
- Pacotes globais (`npm install -g <PACOTE>`):
  - Pasta `C:\Users\USUARIO\AppData\Roaming\npm\node_modules`

**Instalamos um pacote e tem esse monte?**  
(isso tudo são as dependências do pacote archiver)



# Compactando um arquivo

```
JS index.js  x
1  const fs = require('fs');
2  const archiver = require('archiver');
3
4  var saida = fs.createWriteStream('target.zip');
5  var compactador = archiver('zip');
6
7  saida.on('close', () => {
8    | console.log(compactador.pointer() + ' bytes totais');
9  });
10
11  // Grava os arquivos na saída
12  compactador.pipe(saida);
13
14  var arquivo = __dirname + '/index.js';
15  compactador.append(fs.createReadStream(arquivo), {name: 'index.js'});
16
17  compactador.finalize();
```

# Executando

```
node index.js  
434 bytes totais
```

Nome	Data de modificaç...	Tipo	Tamanho
node_modules	14/02/2019 16:56	Pasta de arquivos	
dados_produtos.json	14/02/2019 16:50	Arquivo JSON	1 KB
index.js	14/02/2019 17:11	Arquivo JavaScript	1 KB
package.json	14/02/2019 16:50	Arquivo JSON	1 KB
package-lock.json	14/02/2019 16:50	Arquivo JSON	1 KB
README	14/02/2019 16:50	Arquivo de texto	
saida.txt	14/02/2019 16:50	Arquivo de texto	
target.zip	14/02/2019 16:50	Arquivo ZIP	

target.zip (evaluation copy)

File Commands Tools Favorites Options Help

Add Extract To Test View Delete Find Wizard Info VirusScan Comment

target.zip - ZIP archive, unpacked size 549 bytes

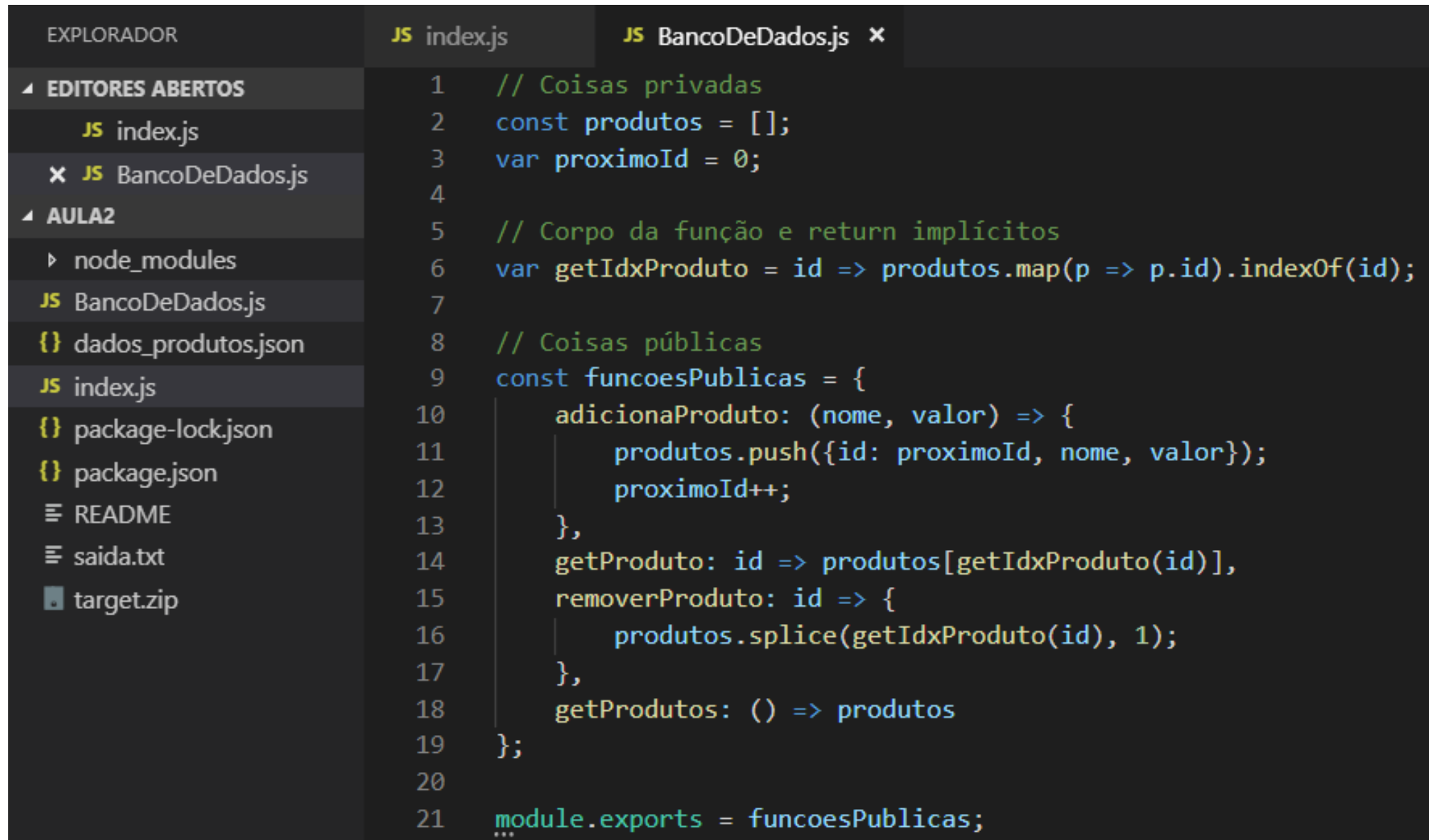
Name	Size	Packed	Type	Modified	CRC32
..			Disco Local		
index.js	549	304	Arquivo JavaScript	14/02/2019 20:10	BD48B6D9

Total 549 bytes in 1 file

# Módulos

- Módulos são arquivos JavaScript ou pastas contendo arquivos;
- Importar módulos globais ou instalados pelo npm:
  - `require(NOME_DO_MODULO);`
- Importar módulos criados pelo usuário (que estão no projeto):
  - `require('CAMINHO_RELATIVO_PARA_O_MODULO');`
- Exemplo:
  - `require('fs');`
  - `require('./util');` // Não precisa colocar a extensão `.js`

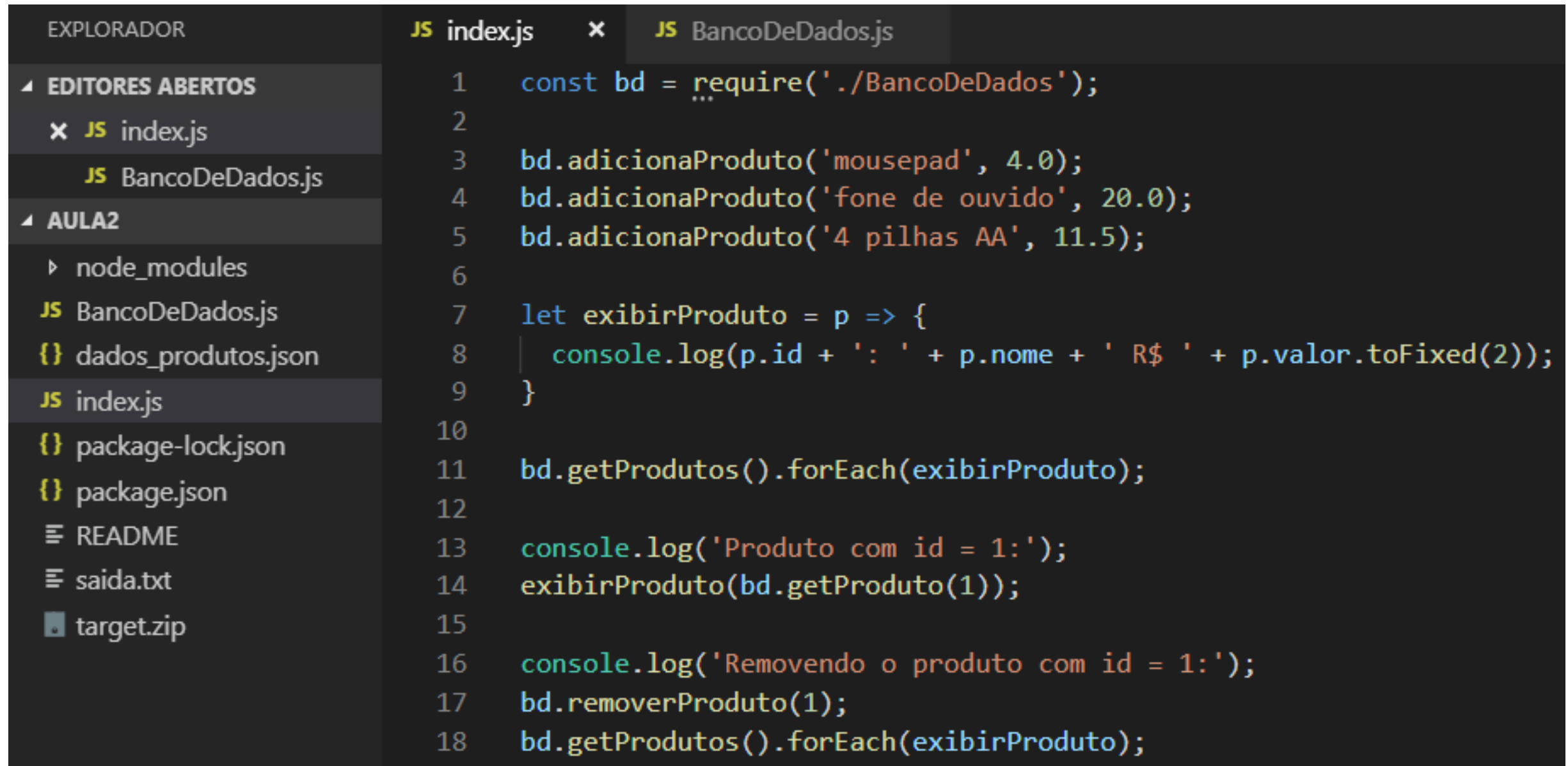
# Novo módulo



The image shows a code editor interface with a dark theme. On the left is a file explorer pane titled 'EXPLORADOR'. It contains a tree view with the following items: 'EDITORES ABERTOS' (expanded), 'index.js' (JS), 'BancoDeDados.js' (JS, with a close icon), 'AULA2' (expanded), 'node\_modules', 'BancoDeDados.js' (JS), 'dados\_produtos.json' (JSON), 'index.js' (JS), 'package-lock.json' (JSON), 'package.json' (JSON), 'README', 'saida.txt', and 'target.zip'. The main editor area shows two tabs: 'index.js' (active) and 'BancoDeDados.js' (with a close icon). The code in 'index.js' is as follows:

```
1 // Coisas privadas
2 const produtos = [];
3 var proximoId = 0;
4
5 // Corpo da função e return implícitos
6 var getIdxProduto = id => produtos.map(p => p.id).indexOf(id);
7
8 // Coisas públicas
9 const funcoesPublicas = {
10   adicionaProduto: (nome, valor) => {
11     produtos.push({id: proximoId, nome, valor});
12     proximoId++;
13   },
14   getProduto: id => produtos[getIdxProduto(id)],
15   removerProduto: id => {
16     produtos.splice(getIdxProduto(id), 1);
17   },
18   getProdutos: () => produtos
19 };
20
21 module.exports = funcoesPublicas;
```

# Usando o módulo criado



EXPLORADOR

EDITORES ABERTOS

- JS index.js
- JS BancoDeDados.js

AULA2

- node\_modules
- JS BancoDeDados.js
- dados\_produtos.json
- JS index.js
- package-lock.json
- package.json
- README
- saida.txt
- target.zip

```
1  const bd = require('./BancoDeDados');
2
3  bd.adicionaProduto('mousepad', 4.0);
4  bd.adicionaProduto('fone de ouvido', 20.0);
5  bd.adicionaProduto('4 pilhas AA', 11.5);
6
7  let exibirProduto = p => {
8    | console.log(p.id + ': ' + p.nome + ' R$ ' + p.valor.toFixed(2));
9  }
10
11 bd.getProdutos().forEach(exibirProduto);
12
13 console.log('Produto com id = 1:');
14 exibirProduto(bd.getProduto(1));
15
16 console.log('Removendo o produto com id = 1:');
17 bd.removerProduto(1);
18 bd.getProdutos().forEach(exibirProduto);
```

# Interface Gráfica para Desktop

- Há duas formas de criar aplicativos com interface gráfica em NodeJS
  1. Utilizar bindings para QT ou GTK:
    - **node-qt** (<https://github.com/arturadib/node-qt>)
    - **node-gui** (<https://github.com/yue/node-gui>)
  2. Frameworks que empacotam aplicações HTML5:
    - **NW.js** (<http://docs.nwjs.io/en/latest/>)
    - **AppJS** (<http://appjs.com/>)
- Há várias aplicações feitas com esses frameworks:
  - *Atom, Visual Studio Code, WhatsApp Desktop, Spotify, Popcorn Time*



# Para a próxima aula

- Valendo pontos na primeira avaliação
  - O projeto valerá 5 pontos
  - Os outros 5 pontos restantes serão um somatório das atividades propostas em sala
- Escreva um aplicativo que compacte uma pasta para um arquivo zip e descompacte um arquivo zip para uma pasta
  - Documentação do pacote archiver:
    - <https://archiverjs.com/docs/#quick-start>
  - Para descompactar, você pode usar o pacote extract-zip:
    - <https://github.com/maxogden/extract-zip>
- Exemplo de uso:
  - `node meu_winrar.js compacta C:\temp\subpasta C:\temp\arquivo.zip`
  - `node meu_winrar.js descompacta C:\temp\arquivo.zip C:\temp\subpasta`