

Tutorial Spring Boot, JPA, Hibernate, H2, Rest, Angular e Boot Strap.

Parte 5

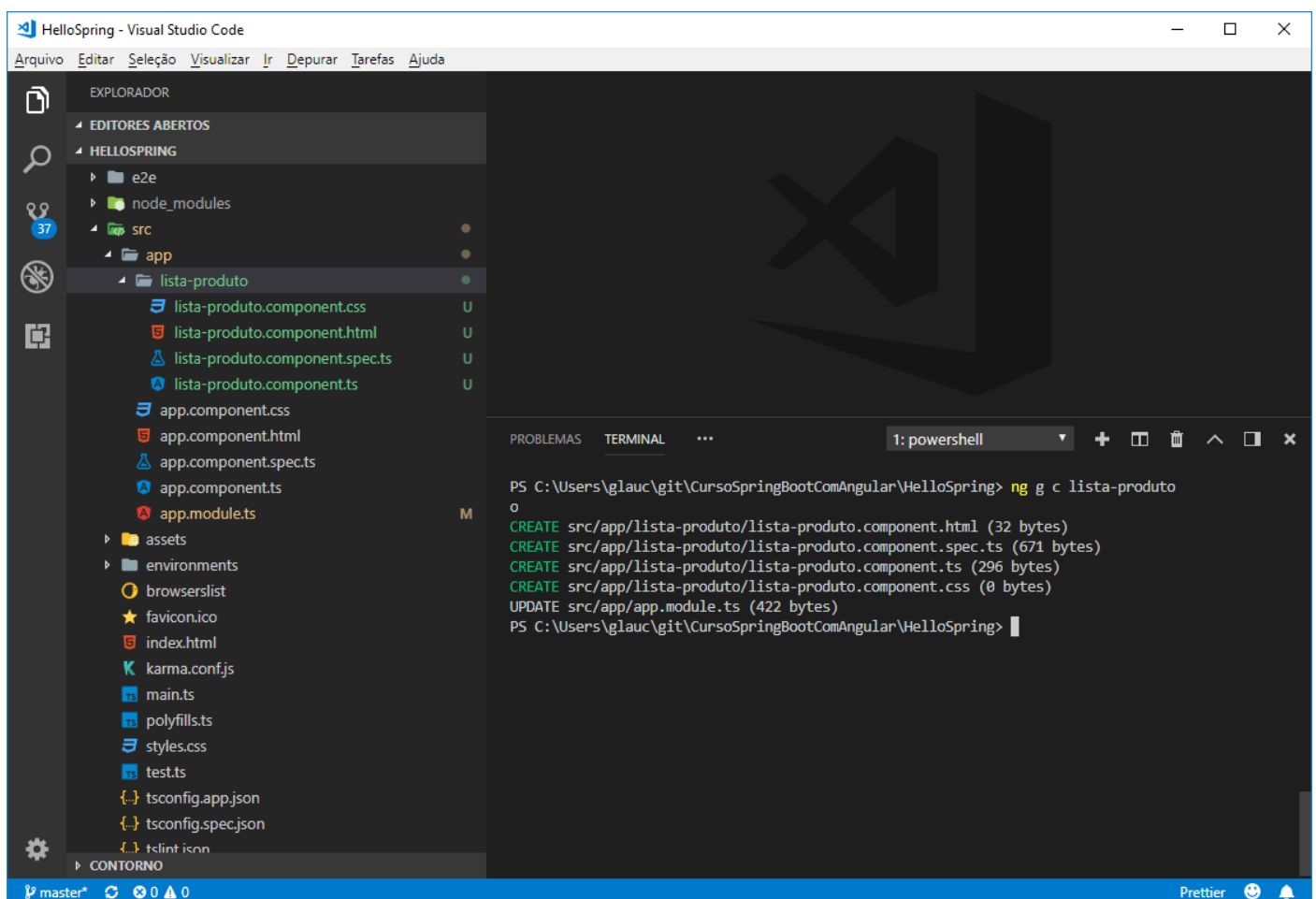
Glauco Todesco

Download:

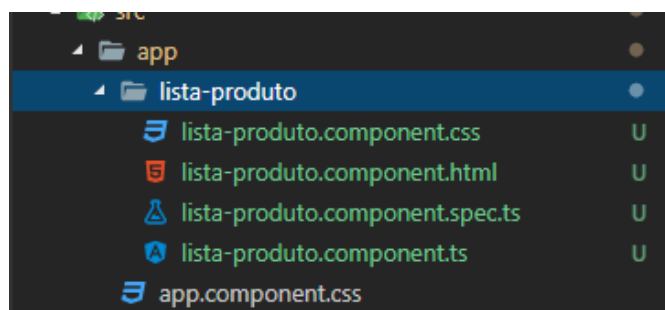
Na quarta parte do tutorial iremos definir inserir com componente para consumir um serviço do webservice.

1. No terminal do VSC digite o comando para criar um componente do angular chamado de lista-produto:

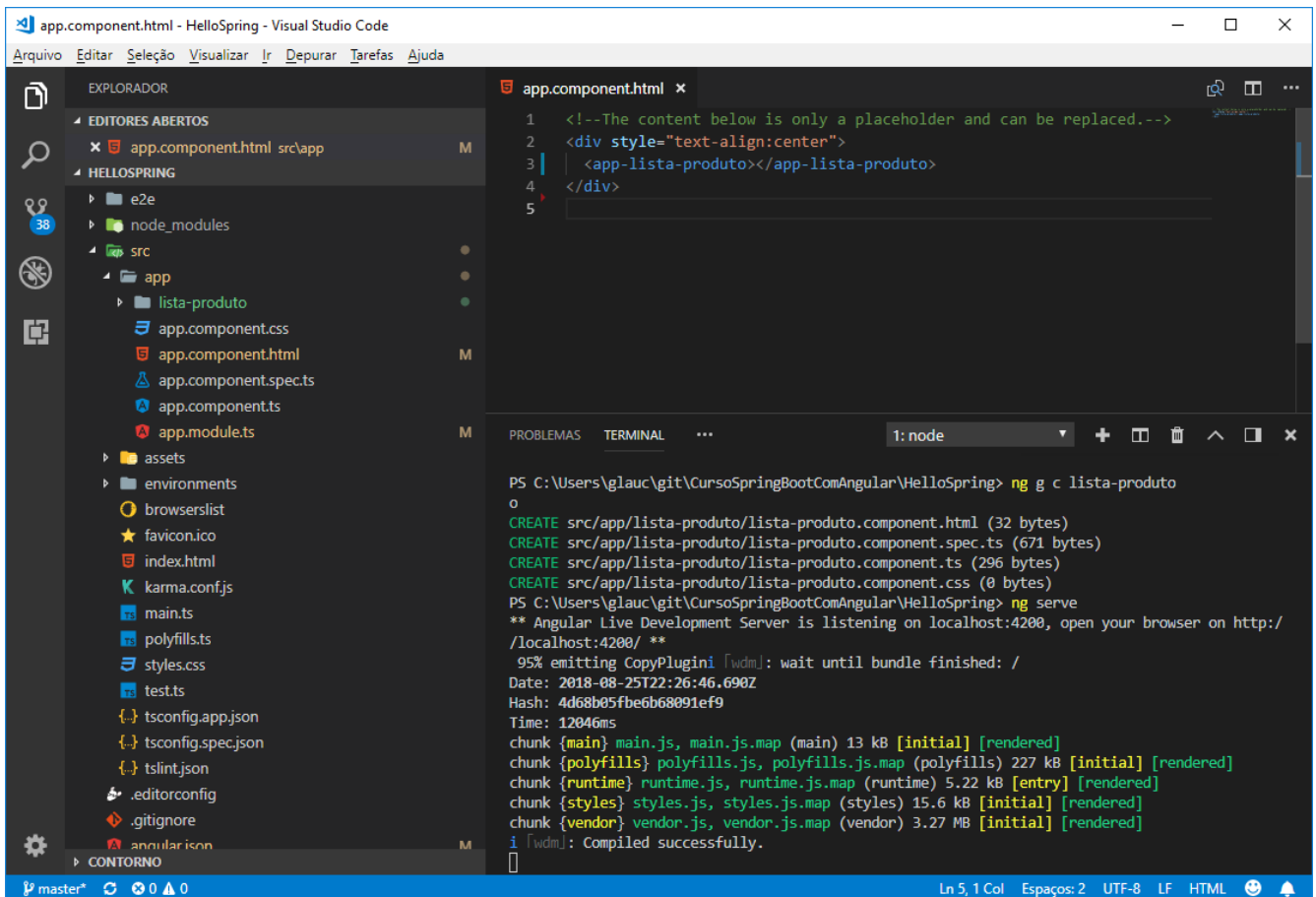
ng g c lista-produto



2. Observe os arquivos criados para esse novo componente:



3. Altere o arquivo `app.component.html` para usar o componente criado, removendo todo o código html existente nesse arquivo e inserido o código do componente criado:



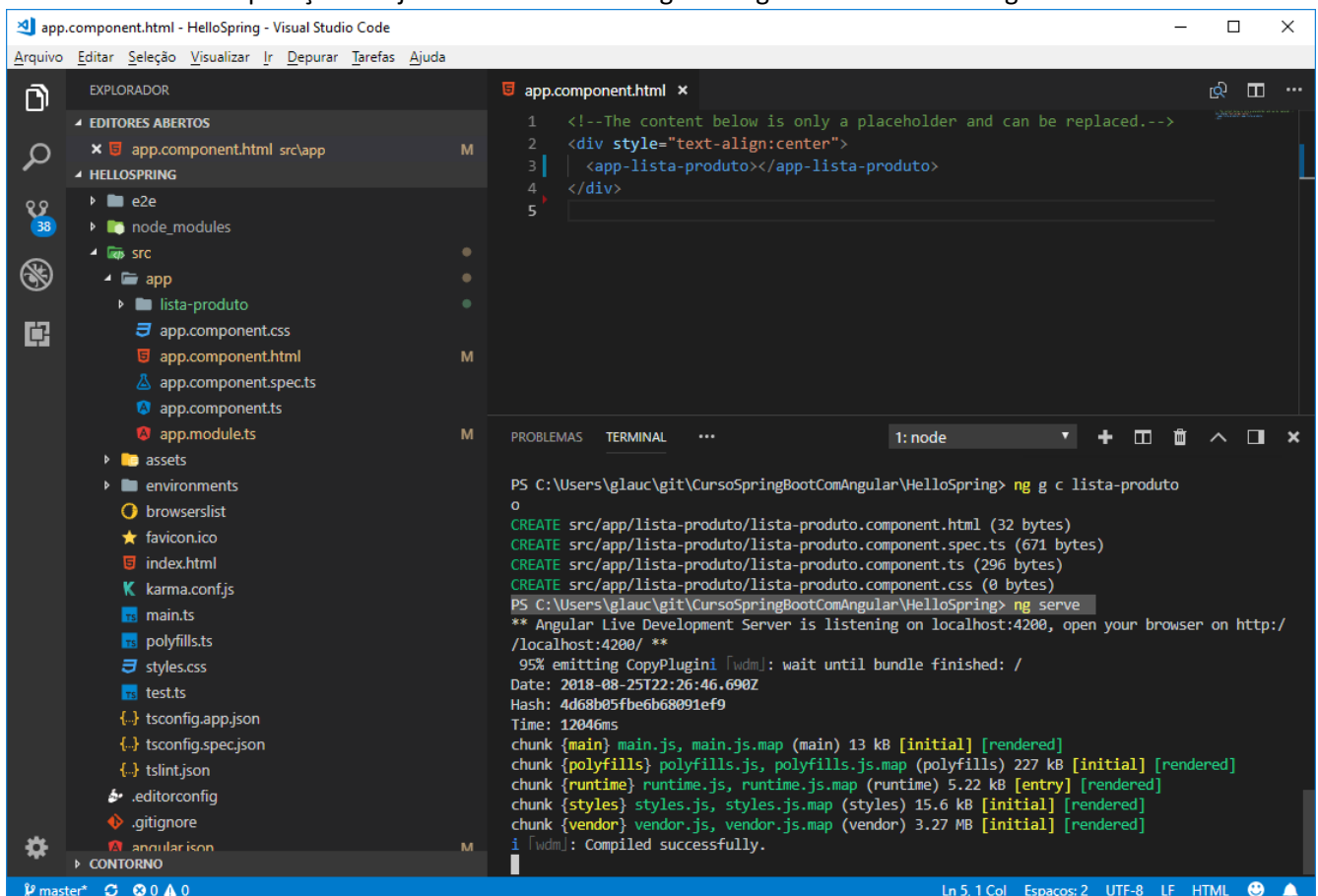
The screenshot shows the Visual Studio Code interface with the `app.component.html` file open. The file content is as follows:

```
1 <!--The content below is only a placeholder and can be replaced.-->
2 <div style="text-align:center">
3   <app-lista-produto></app-lista-produto>
4 </div>
5
```

The Explorer sidebar on the left shows the project structure, including the `src/app` directory and the `lista-produto` component files. The Terminal at the bottom shows the output of the `ng g c lista-produto` and `ng serve` commands:

```
PS C:\Users\glauc\git\CursoSpringBootComAngular\HelloSpring> ng g c lista-produto
o
CREATE src/app/lista-produto/lista-produto.component.html (32 bytes)
CREATE src/app/lista-produto/lista-produto.component.spec.ts (671 bytes)
CREATE src/app/lista-produto/lista-produto.component.ts (296 bytes)
CREATE src/app/lista-produto/lista-produto.component.css (0 bytes)
PS C:\Users\glauc\git\CursoSpringBootComAngular\HelloSpring> ng serve
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
95% emitting CopyPlugin [vdm]: wait until bundle finished: /
Date: 2018-08-25T22:26:46.690Z
Hash: 4d68b05f6e6b68091ef9
Time: 12046ms
chunk {main} main.js, main.js.map (main) 13 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 227 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 5.22 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 15.6 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.27 MB [initial] [rendered]
i [vdm]: Compiled successfully.
```

4. Execute a aplicação e veja o resultado no navegador digitando o comando `ng serve` no terminal:

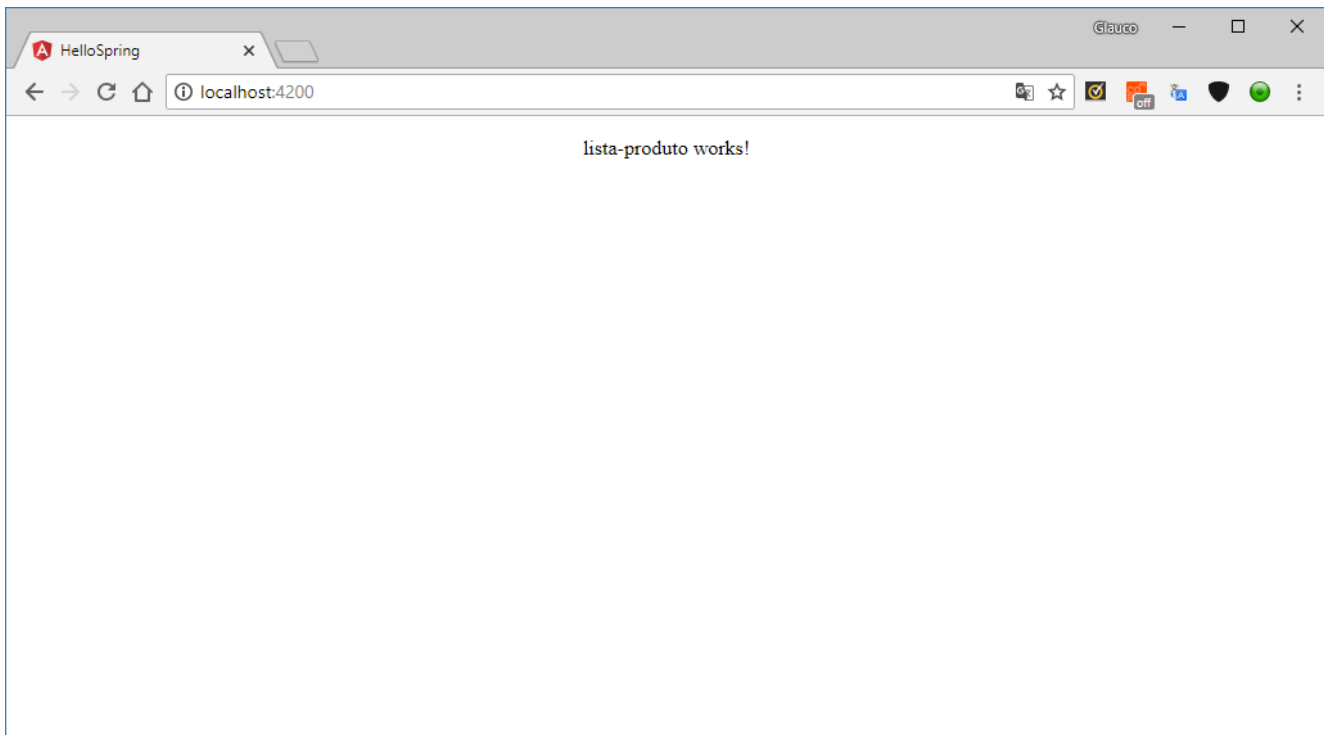


The screenshot shows the Visual Studio Code interface with the `app.component.html` file open. The file content is as follows:

```
1 <!--The content below is only a placeholder and can be replaced.-->
2 <div style="text-align:center">
3   <app-lista-produto></app-lista-produto>
4 </div>
5
```

The Explorer sidebar on the left shows the project structure, including the `src/app` directory and the `lista-produto` component files. The Terminal at the bottom shows the output of the `ng g c lista-produto` and `ng serve` commands:

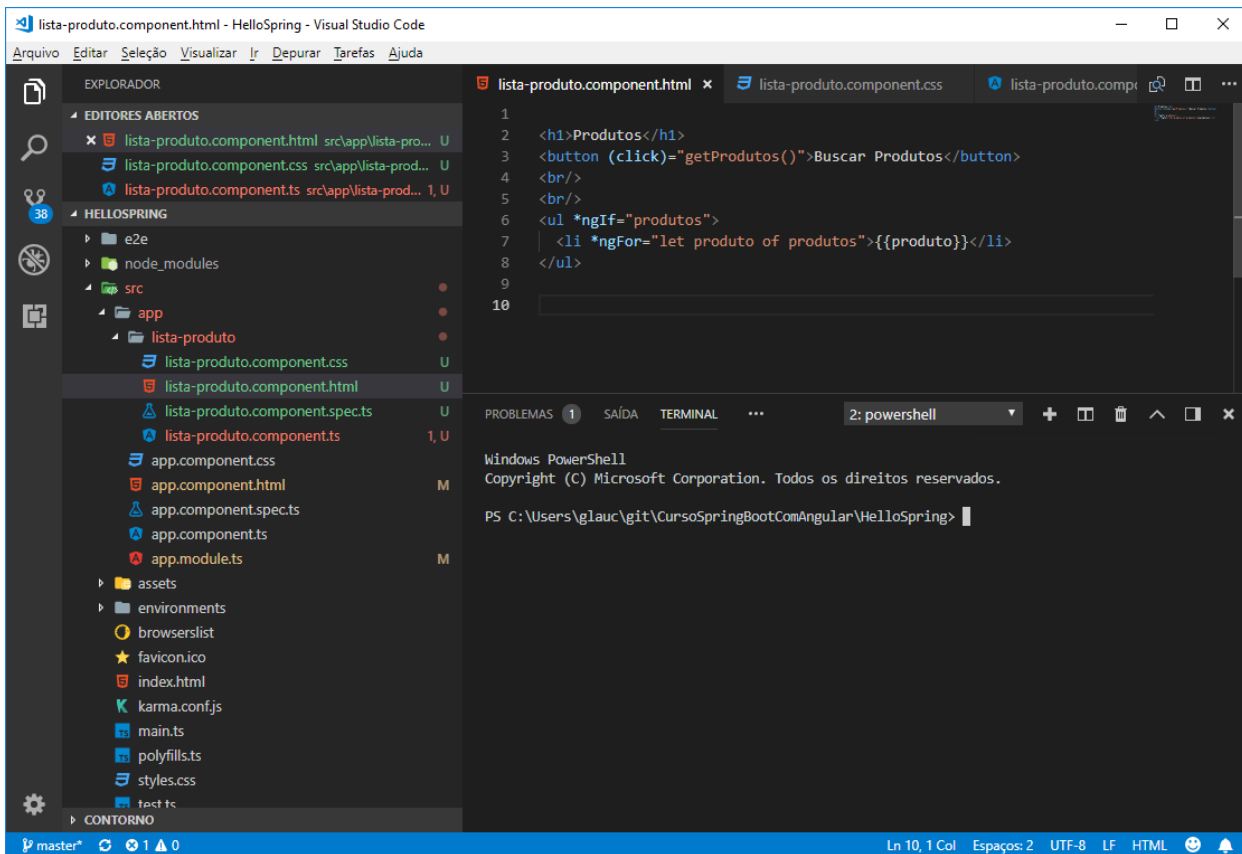
```
PS C:\Users\glauc\git\CursoSpringBootComAngular\HelloSpring> ng g c lista-produto
o
CREATE src/app/lista-produto/lista-produto.component.html (32 bytes)
CREATE src/app/lista-produto/lista-produto.component.spec.ts (671 bytes)
CREATE src/app/lista-produto/lista-produto.component.ts (296 bytes)
CREATE src/app/lista-produto/lista-produto.component.css (0 bytes)
PS C:\Users\glauc\git\CursoSpringBootComAngular\HelloSpring> ng serve
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
95% emitting CopyPlugin [vdm]: wait until bundle finished: /
Date: 2018-08-25T22:26:46.690Z
Hash: 4d68b05f6e6b68091ef9
Time: 12046ms
chunk {main} main.js, main.js.map (main) 13 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 227 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 5.22 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 15.6 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.27 MB [initial] [rendered]
i [vdm]: Compiled successfully.
```



5. Agora vamos implementar um método no nosso componente para retornar todos os produtos cadastrados, inicialmente no modo **hardcode**, sem chamar o nosso webservice. Altere o arquivo lista-produto.component.ts para retornar uma array de produtos. As linhas 10, 17,18 e 19 foram acrescentadas para fazer isso:

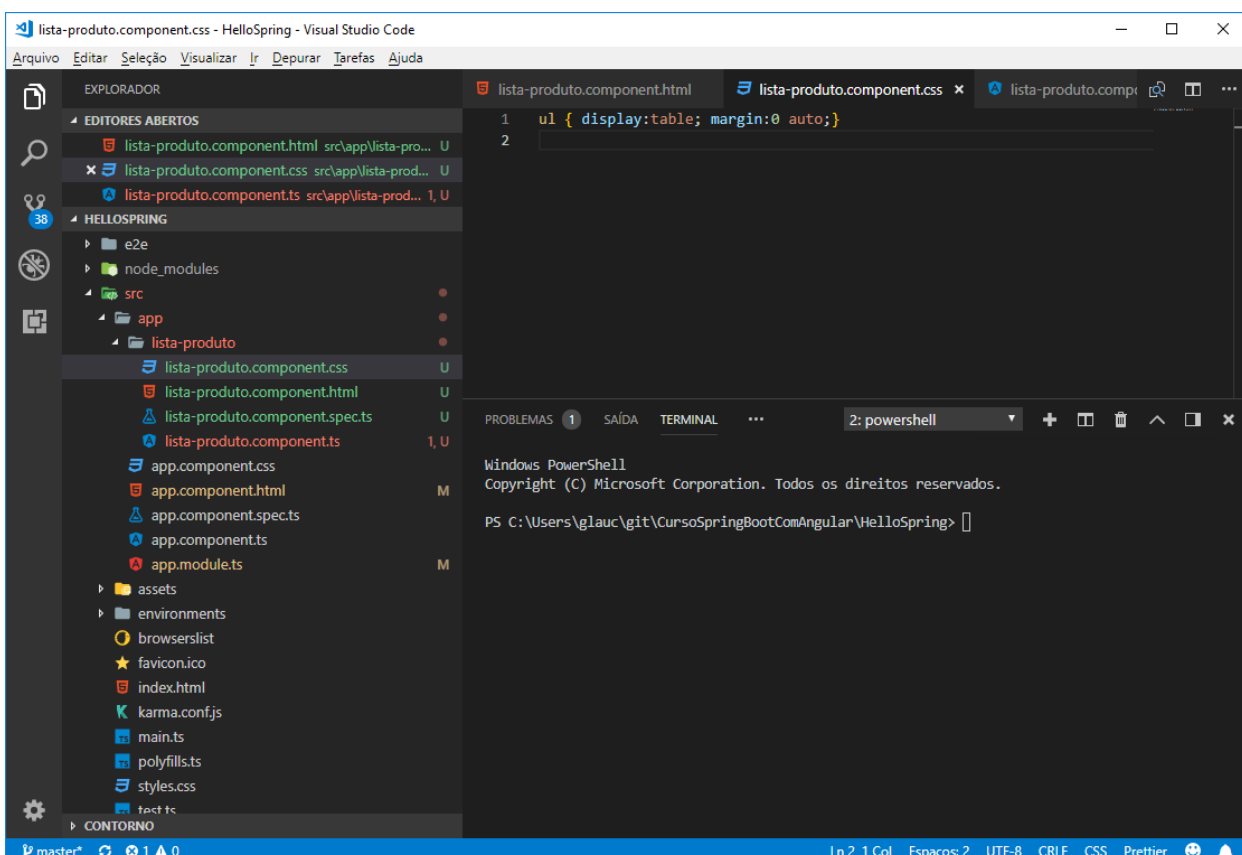
```
lista-produto.component.html  lista-produto.component.css  lista-produto.component.ts x
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-lista-produto',
5    templateUrl: './lista-produto.component.html',
6    styleUrls: ['./lista-produto.component.css']
7  })
8  export class ListaProdutoComponent implements OnInit {
9
10     produtos: string[];
11
12     constructor() { }
13
14     ngOnInit() {
15     }
16
17     getProdutos(){
18       this.produtos = ['Produto 1', 'Produto 2', 'Produto 3'];
19     }
20 }
```

6. Agora vamos alterar o arquivo de template (*.html) para percorrer todo o array e mostrar todos os produtos na tela. No código abaixo o botão chama o método `getProdutos()` que define valores para o array de produtos. Ao fazer isso o `*ngIf` verifica que o array possui dados e através do comando `*ngFor` percorre todos os elementos do array e monta uma lista não ordenada:



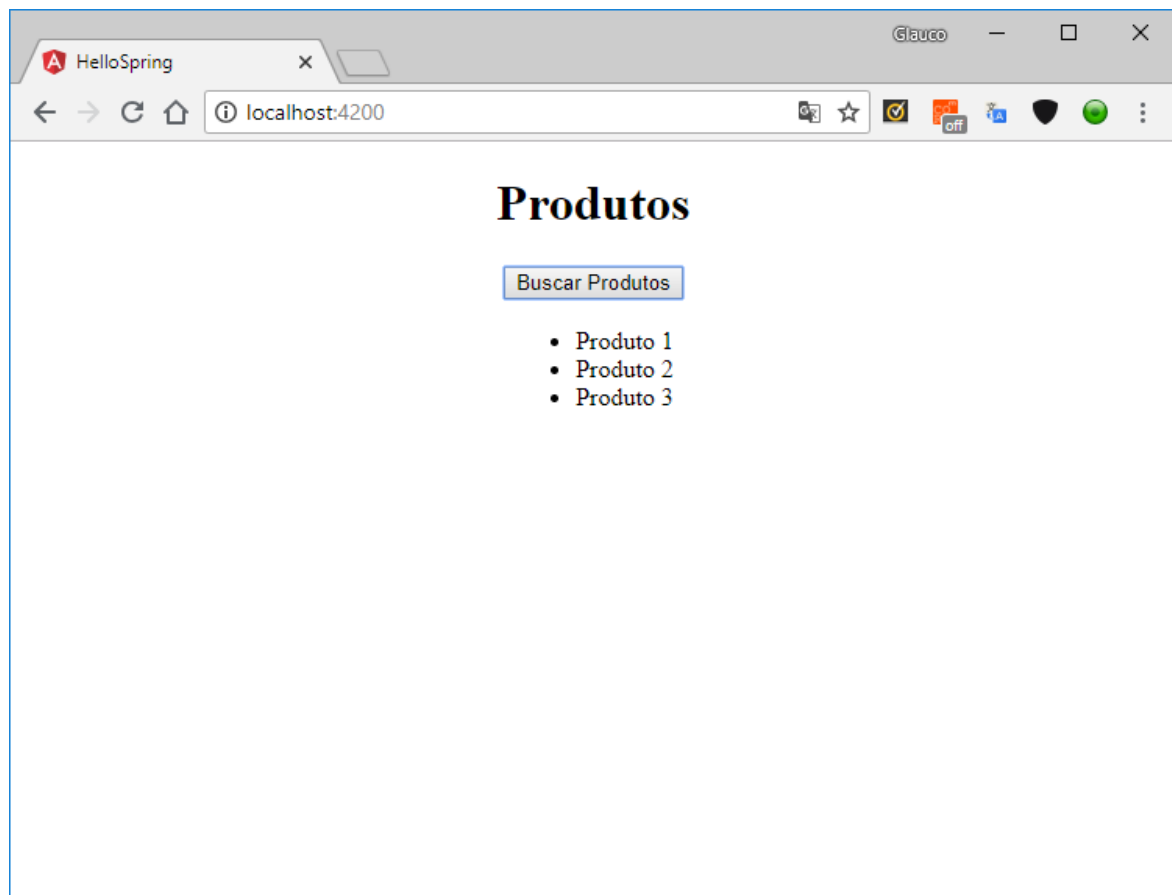
```
1
2 <h1>Produtos</h1>
3 <button (click)="getProdutos()">Buscar Produtos</button>
4 <br/>
5 <br/>
6 <ul *ngIf="produtos">
7   <li *ngFor="let produto of produtos">{{produto}}</li>
8 </ul>
9
10
```

7. Agora vamos alterar o estilo do nosso componente para deixar tudo centralizado alterando o arquivo *.css:



```
1 ul { display:table; margin:0 auto;}
2
```

8. Execute a aplicação e ao clicar no botão teremos o seguinte resultado:



9. Para finalizar faça o build da aplicação (comando **ng build**) e veja ela integrada com o Spring. Observe que a execução pelo Spring é na porta 8080.

