

Implementação de Service de consumo de API

LH-Games

[Parte 1: Preparação da atividade, Validação JSON e Postman](#)

[Parte 2: Serviço API de produtos](#)

[Parte 3: Rotas e CRUD](#)

[Parte 4: Serviço de login, logout e restrição](#)

Nesta atividade, utilizaremos APIs para cadastrar produtos e exibí-los no site da loja de games.

Para cadastrar produtos no site, é necessário realizar o login com usuário e senha. Ao efetuar login, é exibida a página com os jogos já cadastrados. Nela, também, é possível cadastrar produtos novos através do botão *Cadastrar Produto*.

A tela de cadastro possui quatro campos: *Nome do Produto*, *Descrição*, *Foto* e *Preço*. Ao preencher os campos e pressionar o botão *Cadastrar*, em seguida, o sistema emite pop-up, indicando a gravação dos dados.

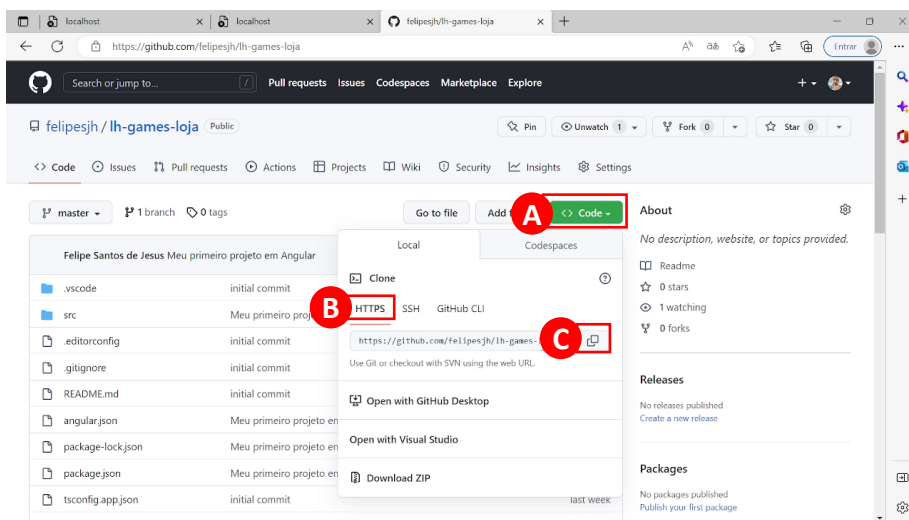
Ainda, é possível excluir um produto ou mesmo editar suas informações.

Vamos iniciar a atividade?

Preparação dos arquivos

Importante!

Você pode usar os arquivos do projeto desenvolvido na atividade da SP1, o qual subiu no GitHub. Para isso, abra seu diretório no **GitHub**, clique em **Code** (A) e, depois, em **HTTPS** (B), conforme indicado abaixo. Utilize o botão de **copiar** (C) para guardar o endereço.



1. Em seu computador, crie uma pasta chamada **loja-angular**. Na barra de endereços do Windows Explorer, digite **cmd** para abrir o Terminal.



2. No Terminal, digite o comando abaixo e pressione **Enter** para que seja efetuada a instalação do projeto. Caso utilize seu próprio projeto do GitHub, substitua somente a área destacada.

```
git clone https://github.com/felipesjh/lh-games-loja.git
```

3. No Terminal, digite o comando abaixo e pressione **Enter** para entrar na pasta.

```
cd lh-games-loja
```

4. Agora, digite o comando abaixo e pressione **Enter** para acessar o NPM.

```
npm install
```

Importante!

O comando **npm install** só é necessário caso o arquivo seja baixado do GitHub. Se estiver realizando o projeto do início, a instalação da biblioteca foi realizada anteriormente.

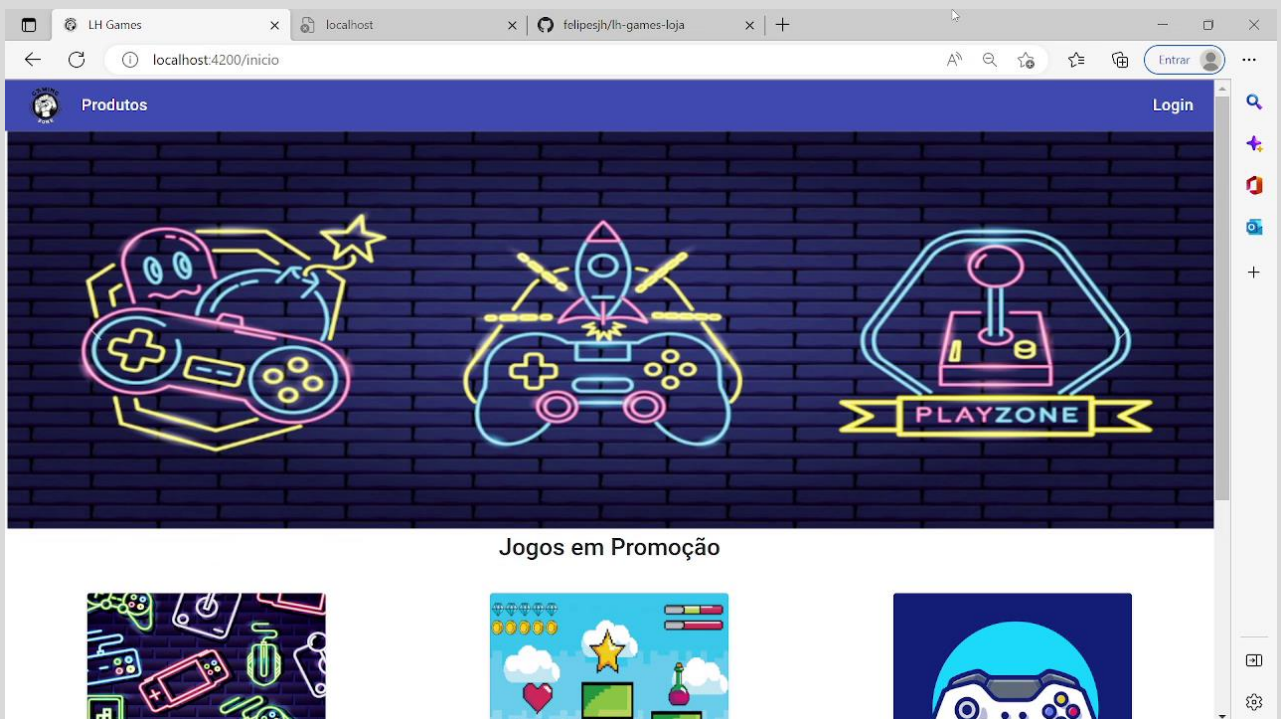


5. No Terminal, digite o comando abaixo e pressione **Enter** para rodar o projeto.

```
ng serve
```

6. Para testar o projeto, abra o navegador e digite a porta padrão (4200) na barra de endereços, para verificar se está funcionando corretamente, conforme abaixo:

```
localhost:4200
```

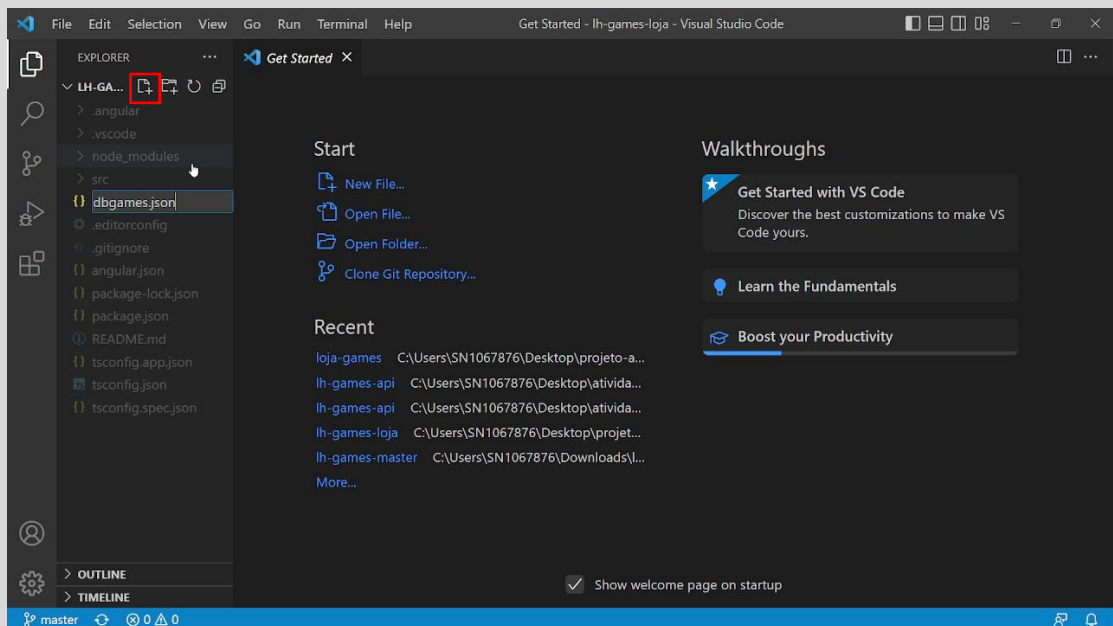


Criando a estrutura de pastas

1. Retorne ao Terminal, digite o comando abaixo e pressione **Enter** para abrir o projeto no VSCode.

```
code .
```

2. Com o VSCode inicializado e com o projeto aberto, crie uma pasta. Para isso, clique no botão indicado abaixo e nomeie como **dbgames.json**.



3. No arquivo **dbgames.json**, digite o código abaixo.

```
{
  "produtos": [
    {
      "id": 1,
      "produto": "Jogo 1",
      "descricao": "Descrição do jogo 1",
      "foto": "jogo1.PNG",
      "preco": 300
    },
    {
      "id": 2,
      "produto": "Jogo 2",
      "descricao": "Descrição do jogo 2",
      "foto": "jogo2.PNG",
      "preco": 200
    },
    {
      "id": 3,
      "produto": "Jogo 3",
      "descricao": "Descrição do jogo 3",
      "foto": "jogo3.PNG",
      "preco": 400
    }
  ]
}
```

Dica!

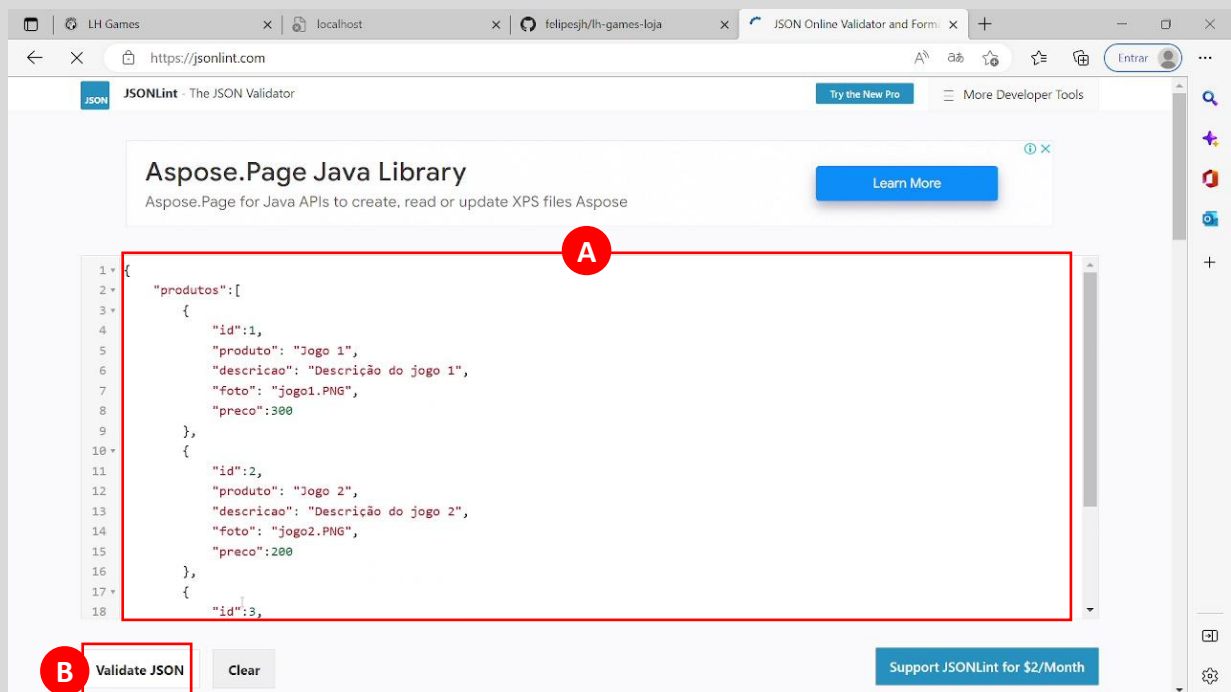
No atributo *foto*, devemos inserir arquivos já salvos na pasta do projeto, localizada na pasta **lh-games-loja\src\assets\img**. Copie somente o nome do arquivo, incluindo a extensão dele.



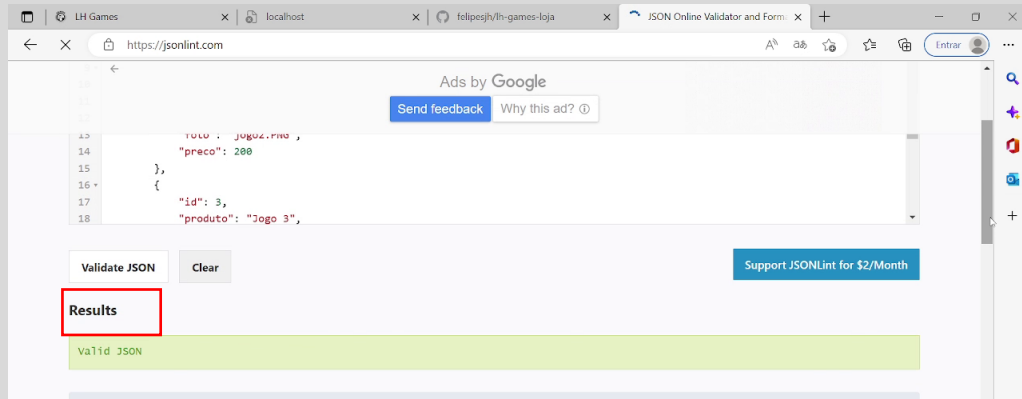
Validação do JSON usando Jsonlint

Para verificar o arquivo JSON que acabamos de digitar, utilizaremos o site Jsonlint.

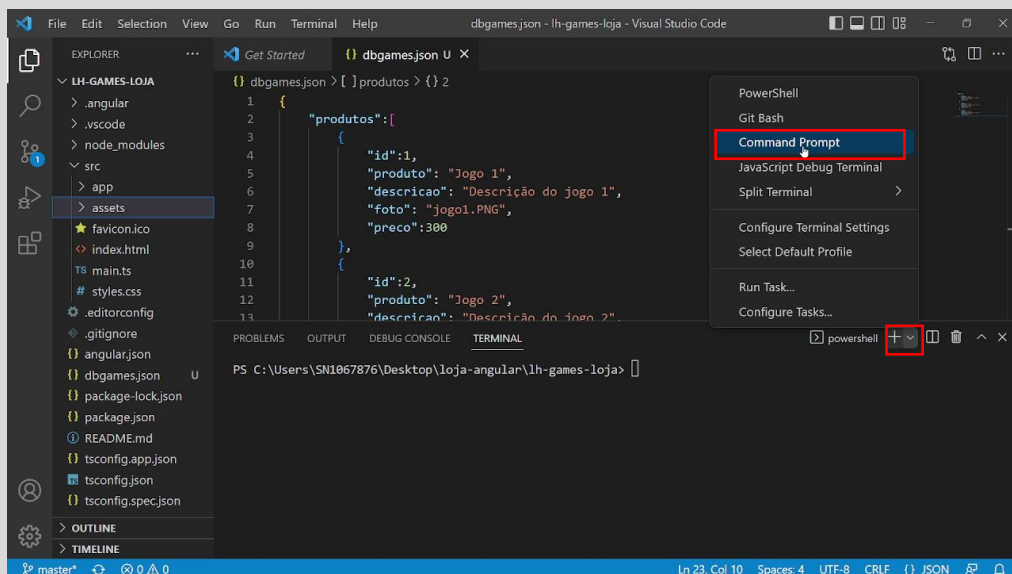
1. No navegador, acesse o link **Jsonlint.com**. Copie o código que digitou no VSCode e cole-o no campo de código (A). Depois, clique em **Validate JSON**, logo abaixo do código digitado (B).



2. Verifique se apareceu a mensagem **Valid JSON** logo abaixo do código. Isso garante a funcionalidade do JSON de nosso código.



3. Volte ao VSCode. No Terminal, na parte inferior do programa, troque a configuração para **Command Prompt**, conforme indicado abaixo.



4. Ainda no Terminal, digite o comando abaixo e dê **Enter** em seguida.

```
npm install -g json-server
```

Dica!

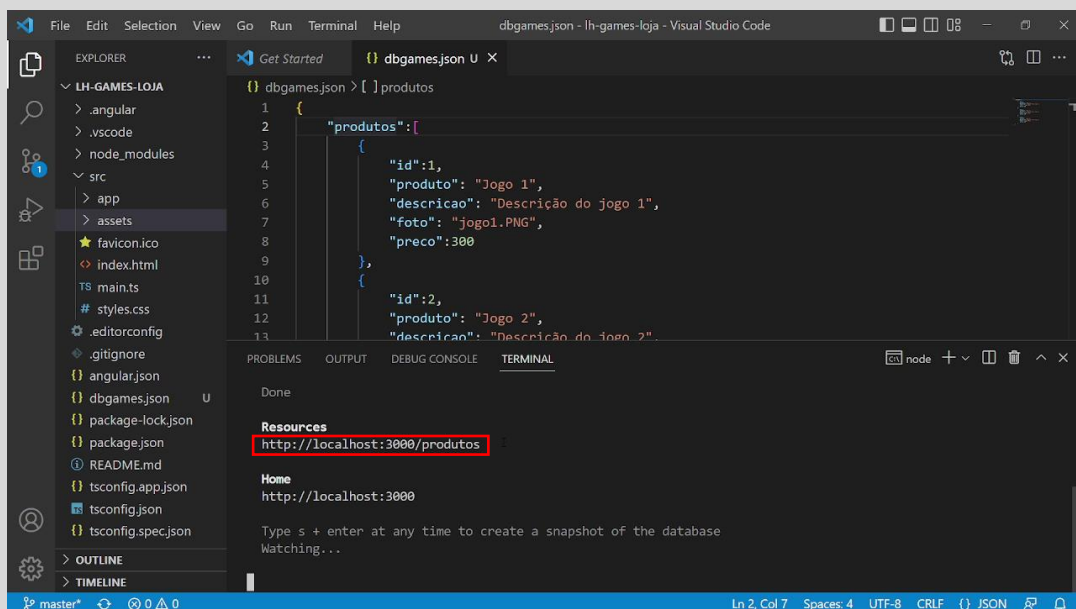
Use o comando **cls** para limpar a tela do Terminal.



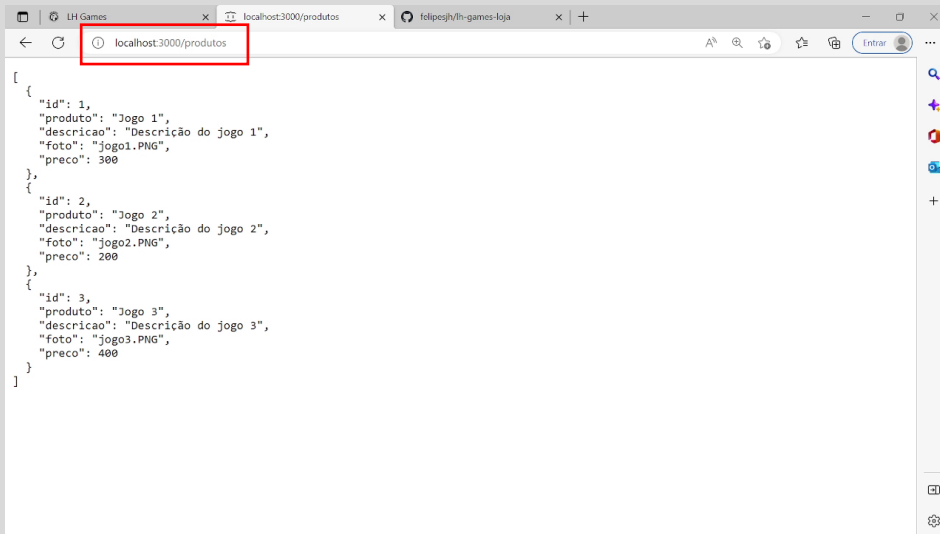
5. Agora, digite o comando abaixo e dê **Enter** em seguida.

```
json-server --watch dbgames.json
```

6. Copie o link gerado no Terminal e indicado abaixo.



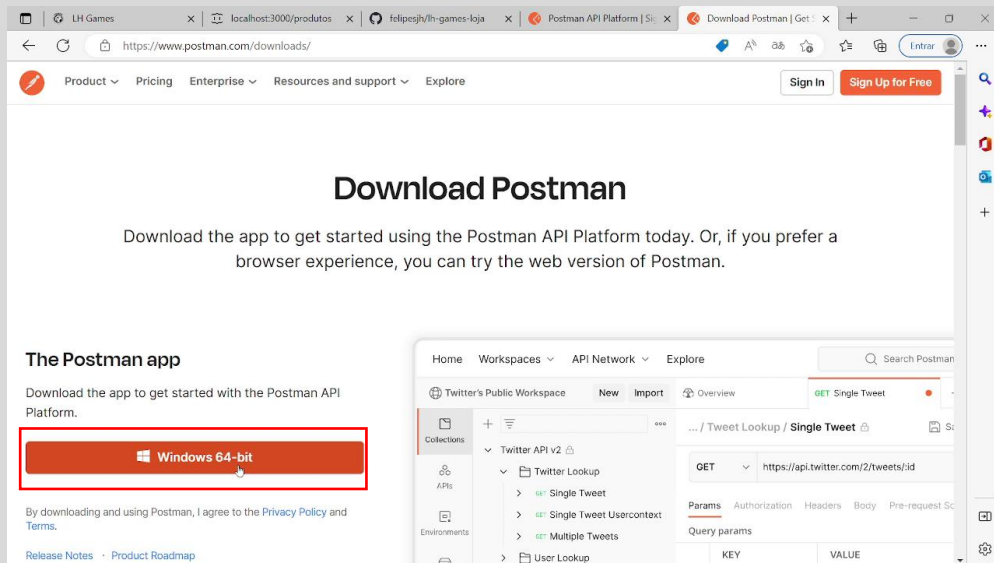
7. Abra o navegador e copie o link gerado no Terminal.



Teste de API para fazer o CRUD

O programa Postman é ideal para fazer testes de funcionamento de API. É um programa gratuito que não exige cadastro caso ele seja instalado no computador.

1. Acesse o site <https://www.postman.com/downloads/>, baixe e realize a instalação padrão do programa.

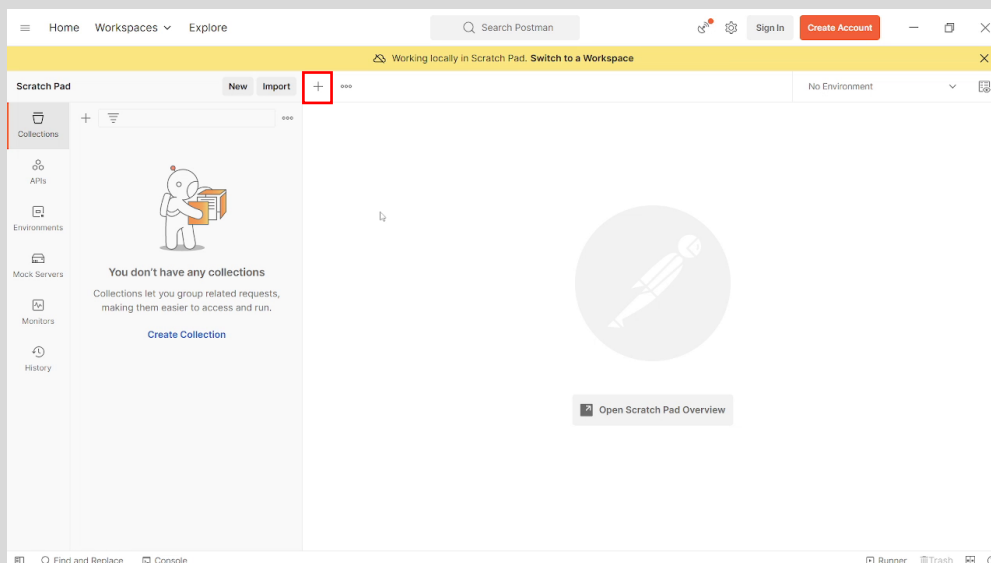


Você sabia?

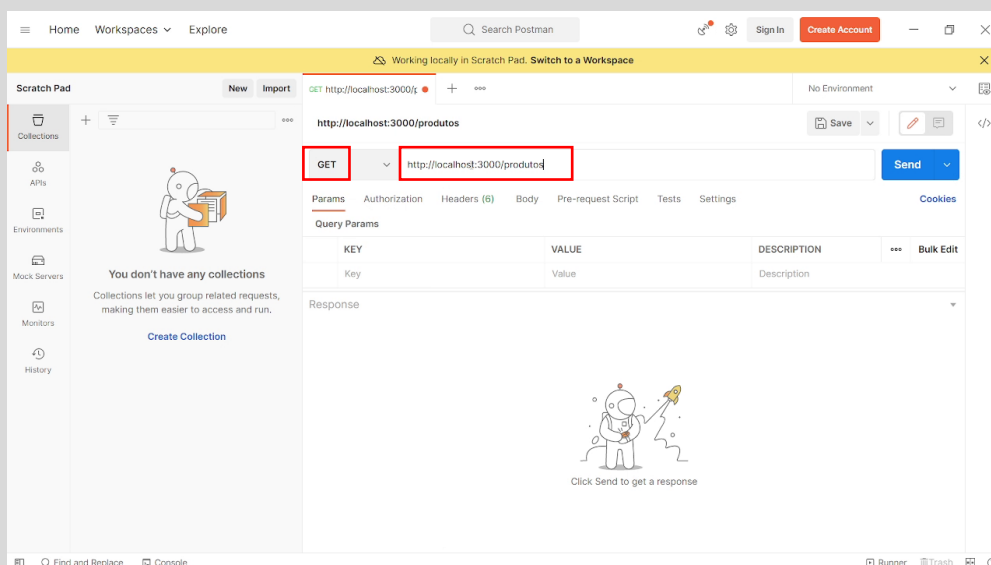
Há uma versão online do Postman. Nesse caso, é necessário realizar cadastro para login na plataforma. Neste tutorial, utilizaremos a versão para download, porém a interface, em ambos os casos, é muito similar.



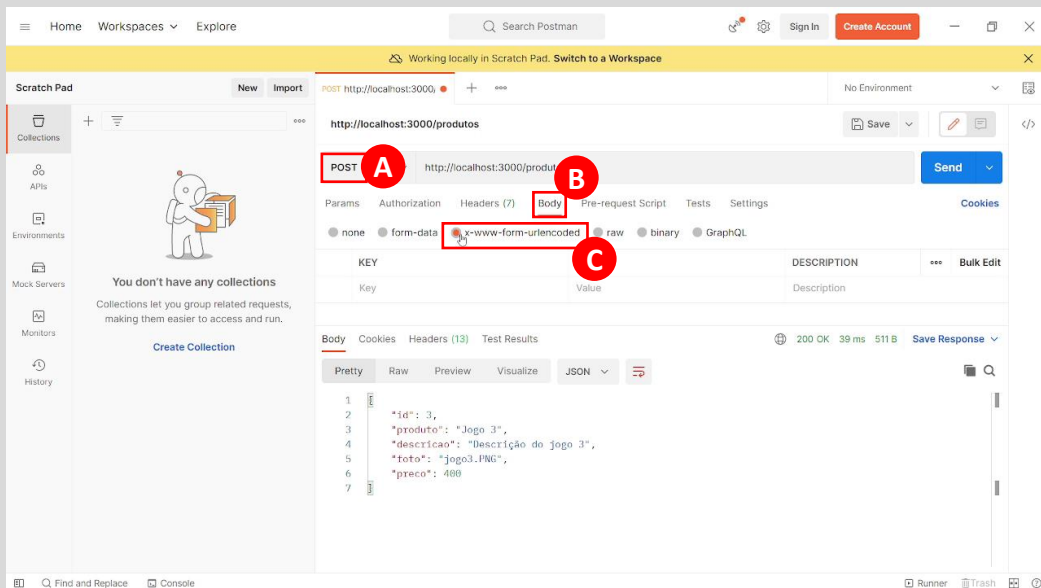
2. Ao abrir o programa, clique no botão “+” na tela principal.



3. Copie o endereço gerado na etapa anterior, incluindo a porta do localhost, e pressione **Send**.



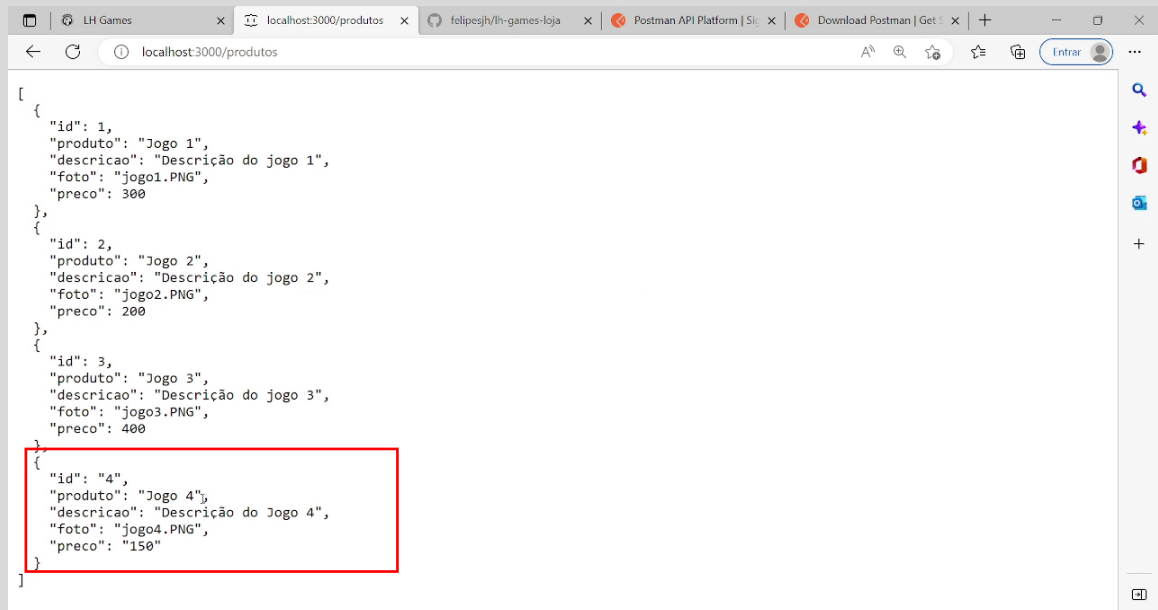
4. Agora, troque a opção GET por **POST** (A), clique na aba **Body** (B) e selecione a opção marcada em (C).



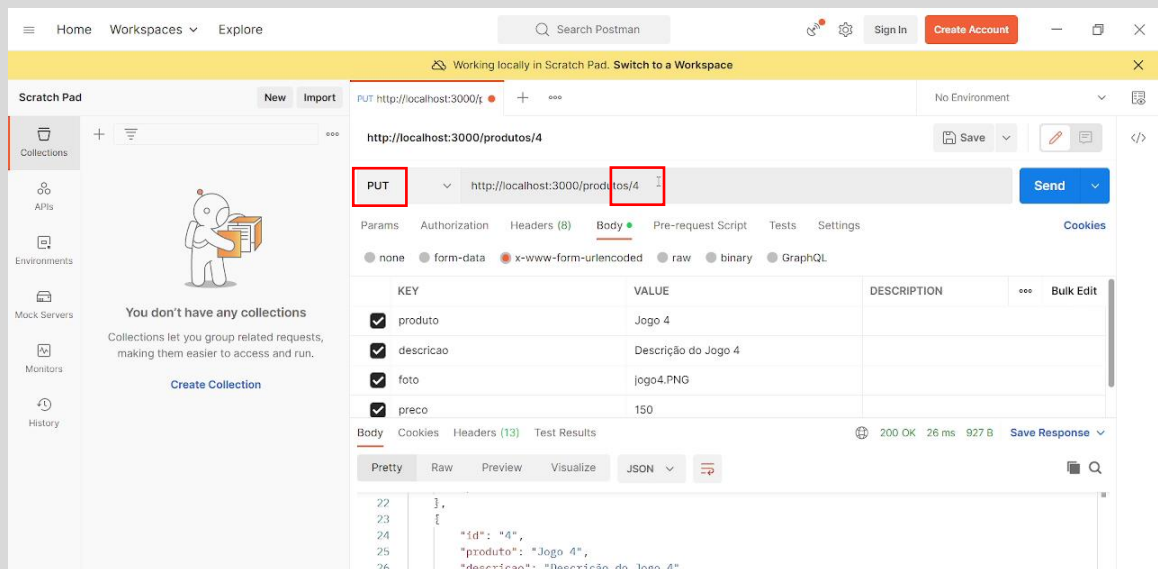
5. Preencha os campos conforme a tabela abaixo e, depois, pressione **SEND** para executar.

KEY	VALUE
id	4
produto	Jogo 4
descricao	Descrição do Jogo 4
foto	jogo.4.PNG
preco	150

6. Ao acessar novamente pelo navegador, o produto Jogo 4 que acabou de ser cadastrado já constará na página.



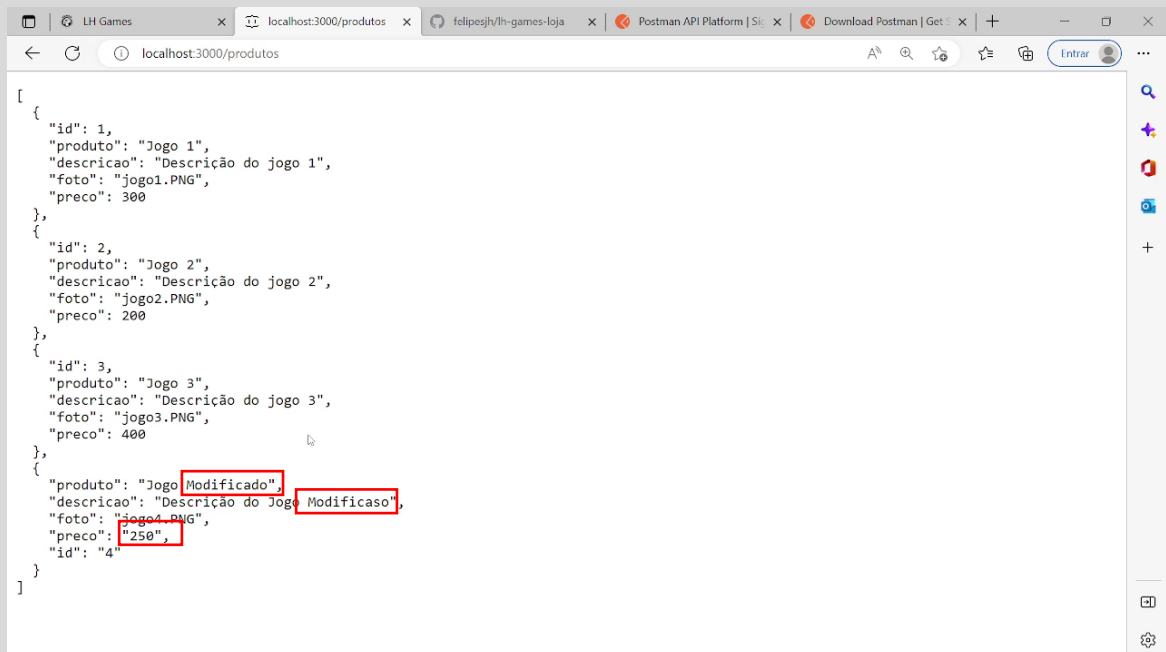
7. Retorne ao programa Post, troque a opção POST por **PUT** e, na barra ao lado do PUT, adicione /4 no final do link exibido. Observe a imagem abaixo.



8. Altere os valores dos atributos, conforme destacado abaixo, e pressione **SEND** para confirmar.

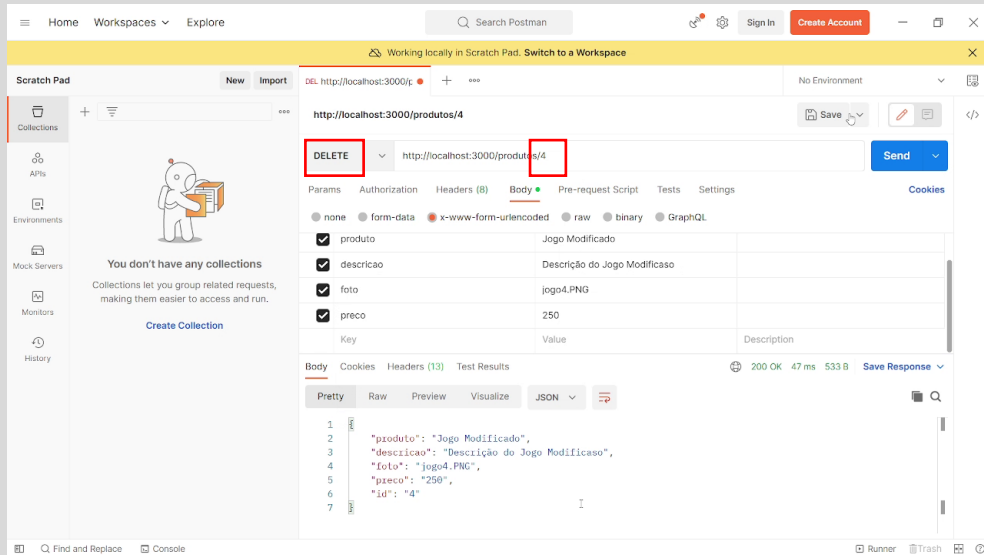
KEY	VALUE
id	4
produto	Jogo Modificado
descricao	Descrição do Jogo Modificado
foto	jogo.4.PNG
preco	250

9. Volte ao navegador e atualize a página para checar as alterações.



```
[
  {
    "id": 1,
    "produto": "Jogo 1",
    "descricao": "Descrição do jogo 1",
    "foto": "jogo1.PNG",
    "preco": 300
  },
  {
    "id": 2,
    "produto": "Jogo 2",
    "descricao": "Descrição do jogo 2",
    "foto": "jogo2.PNG",
    "preco": 200
  },
  {
    "id": 3,
    "produto": "Jogo 3",
    "descricao": "Descrição do jogo 3",
    "foto": "jogo3.PNG",
    "preco": 400
  },
  {
    "produto": "Jogo Modificado",
    "descricao": "Descrição do Jogo Modificado",
    "foto": "jogo.4.PNG",
    "preco": "250",
    "id": "4"
  }
]
```

10. Retorne ao programa Post e troque a opção PUT por **DELETE**. Cerifique-se de que, na barra à direita, consta o /4 ao final do link exibido e pressione **SEND**.



11. Volte ao navegador e atualize a página para realizar teste. Perceba que o item de ID 4 não existe mais.



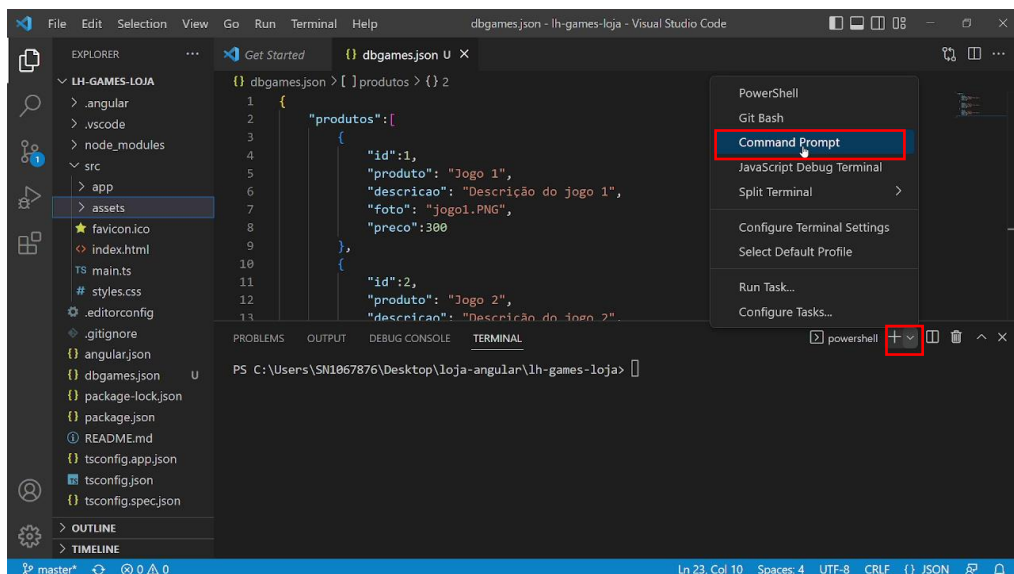
Componentes do projeto

1. Retorne ao VSCode. No Terminal, digite o comando abaixo e dê **Enter** para criar o componente restrito.

```
ng g c inicio  
ng g c login  
ng g c menu  
ng g c rodape  
ng g c restrito
```

Importante!

No terminal, certifique-se de trocar a configuração para **Command Prompt**, conforme indicado abaixo.



2. No Terminal, digite os comandos abaixo, pressionando **Enter** a cada linha, para que sejam criados os outros componentes.

```
ng g c restrito/atualiza-produto  
ng g c restrito/cadastro-produto  
ng g c restrito/lista-produto  
ng g c restrito/menu-restrito
```

3. Abra o arquivo **app.componente.html** e substitua-o pelo código a seguir.

```
<app-menu></app-menu>  
<router-outlet></router-outlet>
```

4. Agora, vamos substituir os códigos dos arquivos html e respectivos css que estão nas pastas localizadas em **src > app**. O processo será realizado nas quatro pastas que acabamos de criar: **atualiza-produto**, **cadastro-produto**, **lista-produto** e **menu-restrito**. Confira a seguir os nomes dos arquivos e os códigos correspondentes que devem ser copiados.

Dica!

Antes de copiá-los, alinhe a indentação dos códigos. Para isso, use o **clique direito** do mouse no código e selecione **Format Document**.



Inicio.component.html

```

<main>
  <section id="section-banner">
    <div id="carouselExampleSlidesOnly" class="carousel slide" data-bs-
ride="carousel">
      <div class="carousel-inner">
        <div class="carousel-item active">
          
        </div>
        <div class="carousel-item">
          
        </div>
        <div class="carousel-item">
          
        </div>
      </div>
    </div>
  </section>

  <mat-grid-list cols="3">
    <mat-grid-tile>
      <mat-card class="example-card" >

        
        <mat-card-header>
          <mat-card-title>Nome produto</mat-card-title>
        </mat-card-header>

        <mat-card-content>
          <p>
            Descrição do Produto
          </p>
          <p>
            Preço do produto<p>
            <button mat-raised-button color="primary">Comprar</button>
          </mat-card-actions>
        </mat-card>
      </mat-grid-tile>
    </mat-grid-list>
  </main>

```

Inicio.component.css

```
#section-banner{
  min-height: 400px;
}

#section-banner img{
  height: 500px;
}

mat-grid-list{
  margin-top: 10px;
  margin-bottom: 10px;
}

mat-grid-tile{
  padding: 2px;
  border-radius: 30px;
}

mat-card-header{
  display: flex;
  align-items: center;
  justify-content: center;
}

mat-card-actions{
  display: flex;
  justify-content: space-around;
}

mat-card-actions p{
  color: red;
  font-size: 30px;
}

mat-card img{
  height: 300px;
  width: 300px;
}

.example-card {
  max-width: 400px;
}

.example-header-image {
  background-image:
url('https://material.angular.io/assets/img/examples/shiba1.jpg');
  background-size: cover;
}

</main>
```

login.component.html

```
<main>
  <section id="section-login">
    <h2>Login</h2>
    <form class="row g-3">

      <label for="id-suário" class="visually-
hidden">Usuário</label>
      <input type="text" class="form-control" id="id-usuario"
placeholder="email@exemplo.com" >

      <br>

      <label for="id-senha" class="visually-
hidden">Senha</label>
      <input type="password" class="form-control" id="id-
senha" placeholder="senha">
      <br>
      <button type="submit" class="btn btn-primary mb-
3">Login</button>

    </form>
  </section>
</main>
```

login.component.css

```
.example-spacer {
  flex: 1 1 auto;
}

main{
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 700px;
}

#section-login{
  height: 100%;
  width: 400px;
  border-radius: 10px;
  border: 2px solid black;
  padding: 10px 20px;
}

span{
  padding: 5px 5px;
}

a{
  text-decoration: none;
  color: white;
}

img{
  height: 50px;
  width: 50px;
}
```

menu.component.html

```
<mat-toolbar color="primary" id="menu">
  <mat-toolbar-row>
    <span><a routerLink="inicio"></a></span>
    <span><a routerLink="inicio">Produtos</a></span>
    <span class="example-spacer"></span>
    <span><a routerLink="login">Login</a></span>
  </mat-toolbar-row>
</mat-toolbar>
```

menu.component.css

```
main{
  display: flex;
  justify-content: center;
  align-items: center;
}

#section-login{
  height: 100%;
  width: 400px;
  border-radius: 10px;
  border: 2px solid black;
  padding: 10px 20px;
  margin-top: 50px;
}
```

rodape.componente.html

```
<footer>
  <p>Desenvolvido por Felipe</p>
</footer>
```

rodape.componente.css

```
footer{
  background: black;
  color: white;
  text-align: center;
  height: 100px;
  padding-top: 30px;
}
```

5. Agora, vamos substituir os códigos dos arquivos html e respectivos css que estão nas pastas localizadas em **src > app > restrito**. Confira a seguir os nomes dos arquivos e os códigos correspondentes que devem ser copiados.

restrito.component.html

```
<app-menu-restrito></app-menu-restrito>
<router-outlet></router-outlet>
```

menu-restrito.component.html

```
<mat-toolbar color="primary">
  <mat-toolbar-row>

    <span><a routerLink="/restrito/lista">Listar Produtos</a></span>
    <span><a routerLink="/restrito/cadastro">Cadastrar Produto</a></span>
    <span class="example-spacer"></span>
    <button mat-icon-button class="example-icon" aria-label="Example icon-button
with share icon">
      <span class="material-icons">logout</span>
    </button>

  </mat-toolbar-row>
</mat-toolbar>
```

menu-restrito.component.html

```
.example-spacer {
  flex: 1 1 auto;
}

span{
  padding: 5px 5px;
}

a{
  text-decoration: none;
  color: white;
}

img{
  height: 50px;
  width: 50px;
}
```


lista-produto.component.html

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">Id</th>
      <th scope="col">Produto</th>
      <th scope="col">Descrição</th>
      <th scope="col">Foto</th>
      <th scope="col">Preço</th>
      <th scope="col">Editar</th>
      <th scope="col">Excluir</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row"></th>
      <td></td>
      <td></td>
      <td></td>
      <td></td>
      <td><a class="btn btn-primary" routerLink="/restrito/editar/"
role="button">Editar</a></td>
      <td><a class="btn btn-danger" role="button">Excluir</a></td>
    </tr>
  </tbody>
</table>
```

lista-produto.component.css

```
table img{
  height: 50px;
  width: 50px;
}
```

cadastro-produto.component.html

```

<main>
  <h1>Cadastrar jogo</h1>
  <section id="section-cadastro">
    <form>

      <label for="produto">Nome do
Produto</label>
      <input type="text" class="form-control"
name="produto">

      <br>

      <label for="descricao">Descrição</label>
      <input type="text" class="form-control"
name="descricao">

      <br>

      <label for="foto">Foto</label>
      <input type="text" class="form-control"
name="foto">

      <br>

      <label for="preco">Preço</label>
      <input type="number" class="form-control"
name="preco">

      <br>

      <button type="button" class="btn btn-
primary mb-3">Cadastrar</button>

    </form>
  </section>
</main>

```

cadastro-produto.component.css

```
main {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}

h1 {
  margin-top: 20px;
}

#section-cadastro {
  height: 100%;
  width: 700px;
  border-radius: 10px;
  border: 2px solid black;
  padding: 10px 10px;
  margin-top: 50px;
  margin-bottom: 10px;
}
```

atualiza-produto.component.html

```
<main>
  <h1>Atualiza jogo</h1>
  <section id="section-cadastro">
    <form>

      <label for="produto">Id</label>
      <input type="text" class="form-control" name="id">

      <br>
      <label for="produto">Nome do Produto</label>
      <input type="text" class="form-control" name="produto">

      <br>

      <label for="descricao">Descrição</label>
      <input type="text" class="form-control"
name="descricao">

      <br>

      <label for="foto">Foto</label>
      <input type="text" class="form-control" name="foto">

      <br>

      <label for="preco">Preço</label>
      <input type="number" class="form-control" name="preco">

      <br>

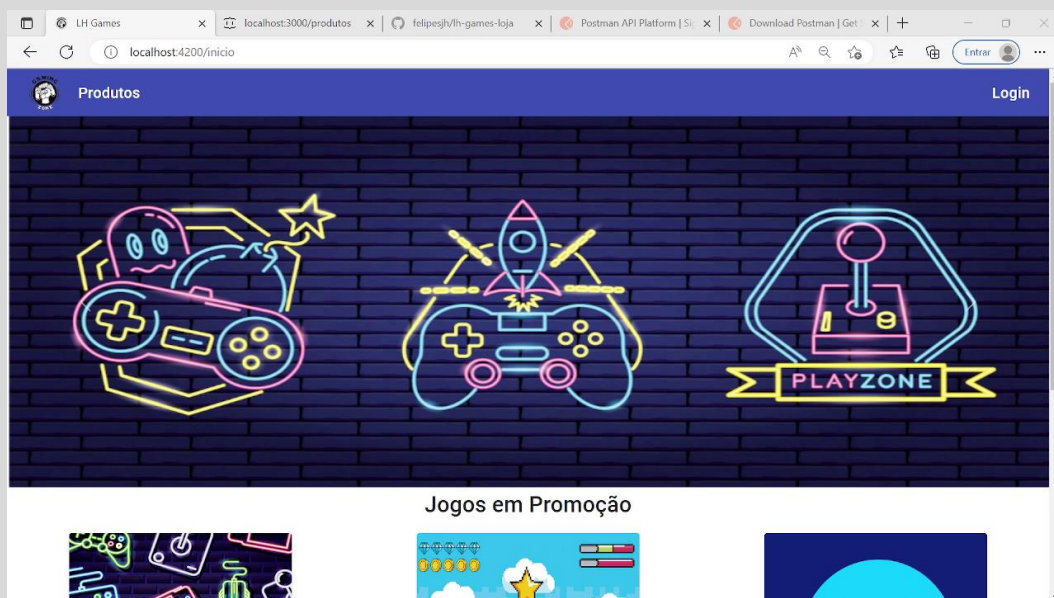
      <button type="submit" class="btn btn-primary mb-
3">Atualizar</button>

    </form>
  </section>
</main>
```

atualiza-produto.component.css

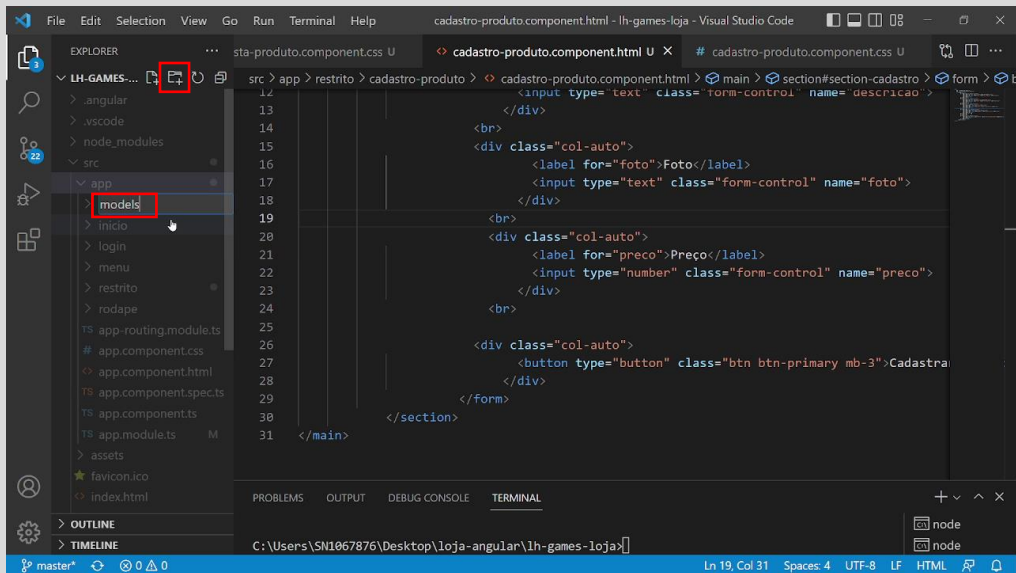
```
main {  
  display: flex;  
  flex-direction: column;  
  justify-content: center;  
  align-items: center;  
}  
  
h1 {  
  margin-top: 20px;  
}  
  
#section-cadastro {  
  height: 100%;  
  width: 700px;  
  border-radius: 10px;  
  border: 2px solid black;  
  padding: 10px 20px;  
  margin-top: 50px;  
  margin-bottom: 10px;  
}
```

6. Salve os arquivos (**Ctrl + S**). Para testar o projeto, abra o navegador e digite o endereço **localhost:4200** na barra de endereços.



Criando a pasta models

1. Voltando ao VSCode, localize a pasta src > app, clique no botão de criar pasta e nomeie como models.



2. Dentro da pasta models, crie um arquivo e nomeie como **Produto.model.ts**. Abra esse arquivo e digite o código abaixo.

```
export class Produto{
  id: number = 0;
  produto: string = "";
  descricao: string = "";
  foto: string = "";
  preco: number = 0;

  constructor(id: number, produto: string, descricao: string, foto: string, preco:
number){
    this.id = id;
    this.produto = produto;
    this.descricao = descricao;
    this.foto = foto;
    this.preco = preco;
  }
}
```

Criando serviço de produto

1. No Terminal do VSCode, digite o código abaixo.

```
ng g s produto
```

2. Foi criado o arquivo **produto.service.ts** em `src > app`. Abra-o e faça alteração conforme o código abaixo.

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Produto } from '../models/Produto.model';

@Injectable({
  providedIn: 'root'
})
export class ProdutoService {

  private url = "http://localhost:3000/produtos";

  constructor(private _httpClient: HttpClient) { }

  getProduto(id:any): Observable<Produto> {
    const urlAtualizar = `${this.url}?id=${id}`;
    return this._httpClient.get<Produto>(urlAtualizar);
  }

  getProdutos(): Observable<Produto[]> {
    return this._httpClient.get<Produto[]>(this.url);
  }

  cadastrarProduto(produto: Produto): Observable<Produto[]> {
    return this._httpClient.post<Produto[]>(this.url, produto);
  }

  atualizarProduto(id: any, produto: Produto): Observable<Produto[]> {
    const urlAtualizar = `${this.url}/${id}`;
    return this._httpClient.put<Produto[]>(urlAtualizar, produto);
  }

  removerProduto(id: any): Observable<Produto[]> {
    const urlDeletar = `${this.url}/${id}`;
    return this._httpClient.delete<Produto[]>(urlDeletar);
  }
}
```

Criando rotas

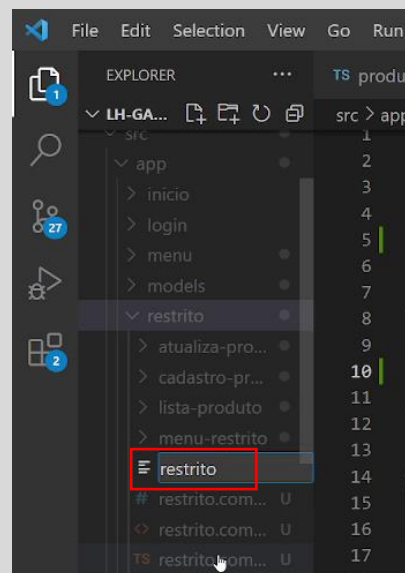
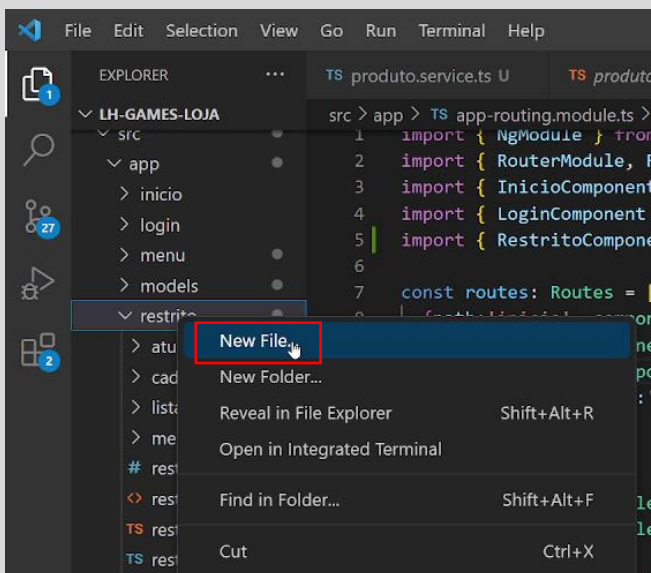
1. Localize o arquivo **app-routing.module.ts** que está na pasta **src > app**. Abra-o e faça alteração conforme o código abaixo.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { InicioComponent } from '../inicio/inicio.component';
import { LoginComponent } from '../login/login.component';
import { RestritoComponent } from '../restrito/restrito.component';

const routes: Routes = [
  {path: 'inicio', component: InicioComponent},
  {path: 'login', component: LoginComponent},
  {path: 'restrito', component: RestritoComponent},
  {path: '', redirectTo: '/inicio', pathMatch: 'full'}
]

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

2. No VSCode, clique no nome da pasta **restrito** com o botão direito e, depois, em **New File**. Nomeie o arquivo como **restrito-routing.module.ts**.



3. Abra o arquivo **restrito-routing.module.ts** e faça alteração conforme o código abaixo.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { InicioComponent } from '../inicio/inicio.component';
import { LoginComponent } from '../login/login.component';
import { RestritoComponent } from '../restrito/restrito.component';

const routes: Routes = [
  {path: 'inicio', component: InicioComponent},
  {path: 'login', component: LoginComponent},
  {path: 'restrito', component: RestritoComponent},
  {path: '', redirectTo: '/inicio', pathMatch: 'full'}
]

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

4. Abra o arquivo **app.module.ts** e faça a substituição de acordo com o código abaixo (continua na próxima página).

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

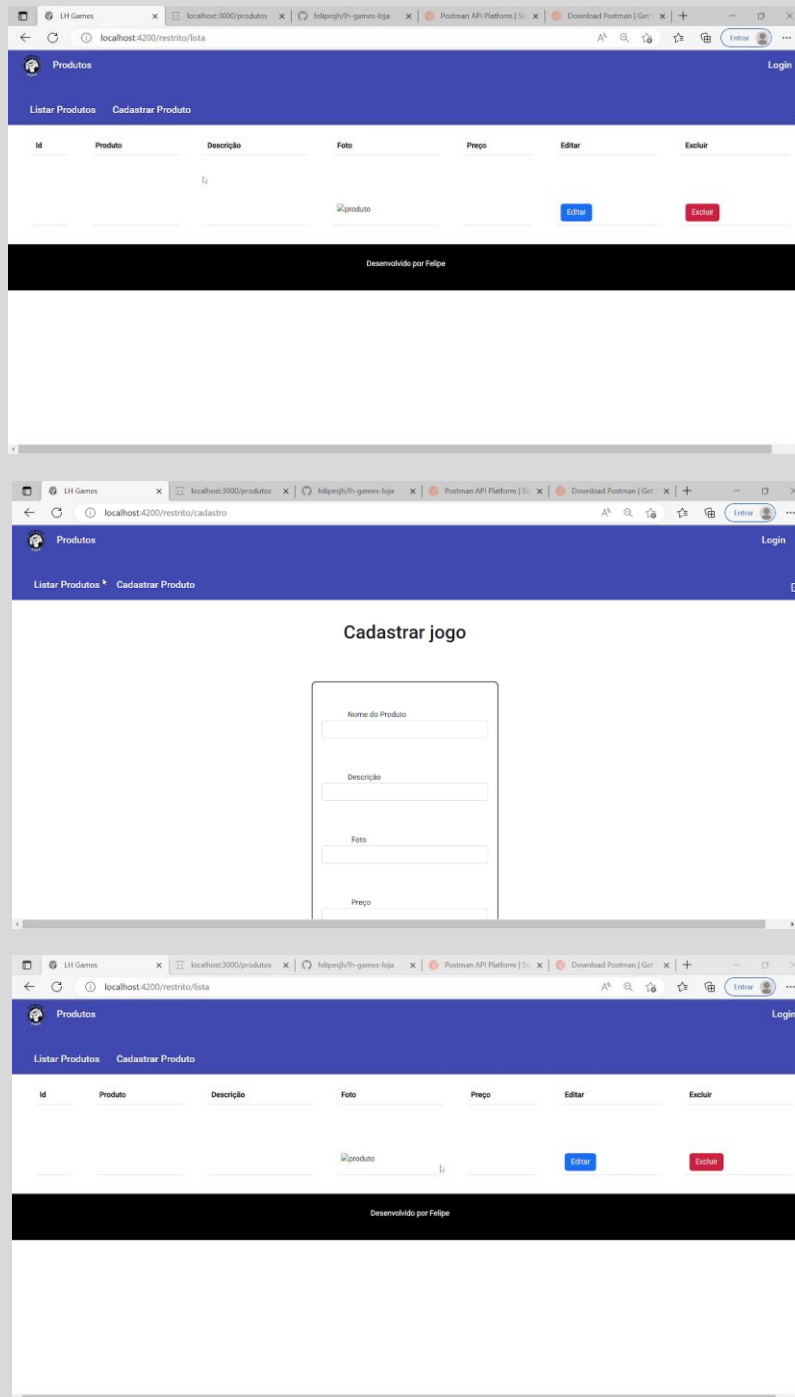
/*Importações Angular Material*/
import { MatButtonModule } from '@angular/material/button';
import { MatCardModule } from '@angular/material/card';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatGridListModule } from '@angular/material/grid-list';
import { MatIconModule } from '@angular/material/icon';
import { MatInputModule } from '@angular/material/input';
import { MatMenuModule } from '@angular/material/menu';
import { MatToolbarModule } from '@angular/material/toolbar';
```

Continuação do código

```
import { AppRoutingModuleModule } from './app-routing.module';
import { RestritoRoutingModule } from './restrito/restrito-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { InicioComponent } from './inicio/inicio.component';
import { LoginComponent } from './login/login.component';
import { MenuComponent } from './menu/menu.component';
import { RodapeComponent } from './rodape/rodape.component';
import { RestritoComponent } from './restrito/restrito.component';
import { AtualizaProdutoComponent } from './restrito/atualiza-produto/atualiza-produto.component';
import { CadastroProdutoComponent } from './restrito/cadastro-produto/cadastro-produto.component';
import { ListaProdutoComponent } from './restrito/lista-produto/lista-produto.component';
import { MenuRestritoComponent } from './restrito/menu-restrito/menu-restrito.component';

@NgModule({
  declarations: [
    AppComponent,
    InicioComponent,
    LoginComponent,
    MenuComponent,
    RodapeComponent,
    RestritoComponent,
    AtualizaProdutoComponent,
    CadastroProdutoComponent,
    ListaProdutoComponent,
    MenuRestritoComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModuleModule,
    BrowserAnimationsModule,
    MatButtonModule,
    MatCardModule,
    MatFormFieldModule,
    MatGridListModule,
    MatIconModule,
    MatInputModule,
    MatMenuModule,
    MatToolbarModule,
    RestritoRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

5. Para testar, abra o navegador e teste o endereço **localhost:4200/restrito/lista** (Caso esteja utilizando outra porta, faça a substituição pelo número correspondente). Teste, também, o menu **Cadastrar Produtos** e **Listar produtos**.



Funcionalidades editar e excluir

1. Copie os códigos a seguir nos arquivos indicados da pasta **Src > app > inicio**.

inicio.component.ts (está na pasta **Src > app > inicio**)

```
import { Component, OnInit } from '@angular/core';
import { Produto } from '../models/Produto.model';
import { ProdutoService } from '../produto.service';

@Component({
  selector: 'app-inicio',
  templateUrl: './inicio.component.html',
  styleUrls: ['./inicio.component.css']
})
export class InicioComponent implements OnInit{

  public produtos: Produto[] = [];

  constructor(private _produtoService: ProdutoService){}

  ngOnInit():void{
    this.listarProdutos();
  }

  listarProdutos():void{
    this._produtoService.getProdutos()
      .subscribe(
        retornaProduto => {
          this.produtos = retornaProduto.map(
            item => {
              return new Produto(
                item.id,
                item.produto,
                item.descricao,
                item.foto,
                item.preco
              );
            }
          );
        }
      )
  }
}
```

inicio.component.html (está na pasta Src > app > inicio)

```

<main>
  <section id="section-banner">
    <div id="carouselExampleSlidesOnly" class="carousel slide" data-bs-
ride="carousel">
      <div class="carousel-inner">
        <div class="carousel-item active">
          
        </div>
        <div class="carousel-item">
          
        </div>
        <div class="carousel-item">
          
        </div>
      </div>
    </div>
  </section>

  <mat-grid-list cols="3">
    <mat-grid-tile *ngFor="let produto of produtos">
      <mat-card class="example-card" >
        
        <mat-card-header>
          <mat-card-title>{{produto.produto}}</mat-card-title>
        </mat-card-header>

        <mat-card-content>
          <p>
            {{produto.descricao}}
          </p>
          <p>
          </mat-card-content>
        <mat-card-actions>
          <p>{{produto.preco | currency : 'BRL'}}</p>
          <button mat-raised-button color="primary">Comprar</button>
        </mat-card-actions>
      </mat-card>
    </mat-grid-tile>
  </mat-grid-list>

</main>

```

lista-produto.component.ts (está na pasta Pasta src > app> restrito > lista-produto)

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { Produto } from 'src/app/models/Produto.model';
import { ProdutoService } from 'src/app/produto.service';

@Component({
  selector: 'app-lista-produto',
  templateUrl: './lista-produto.component.html',
  styleUrls: ['./lista-produto.component.css']
})

export class ListaProdutoComponent {
  public produtos: Produto[] = [ ];

  constructor(private _produtoService: ProdutoService,
    private router: Router){}

  ngOnInit():void{
    this.listarProdutos();
  }

  listarProdutos():void{
    this._produtoService.getProdutos()
      .subscribe(
        retornaProduto => {
          this.produtos = retornaProduto.map(
            item => {
              return new Produto(
                item.id,
                item.produto,
                item.descricao,
                item.foto,
                item.preco
              );
            }
          );
        }
      )
  }

  excluir(id: number){
    this._produtoService.removerProduto(id).subscribe(
      produto => {
        this.listarProdutos();
      },
      err => {console.log("erro ao Excluir")}
    );

    // window.location.href = "/restrito/lista";
    this.router.navigate(["/restrito/lista"]);
  }
}
```

lista-produto.component.html (está na pasta `src > app > restrito > lista-produto`)

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">Id</th>
      <th scope="col">Produto</th>
      <th scope="col">Descrição</th>
      <th scope="col">Foto</th>
      <th scope="col">Preço</th>
      <th scope="col">Editar</th>
      <th scope="col">Excluir</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let p of produtos">
      <th scope="row">{{p.id}}</th>
      <td>{{p.produto}}</td>
      <td>{{p.descricao}}</td>
      <td></td>
      <td>{{p.preco | currency : 'BRL'}}</td>
      <td><a class="btn btn-primary"
routerLink="/restrito/editar/{{p.id}}" role="button">Editar</a></td>
      <td><a class="btn btn-danger" (click)="excluir(p.id)"
role="button">Excluir</a></td>
    </tr>
  </tbody>
</table>
```

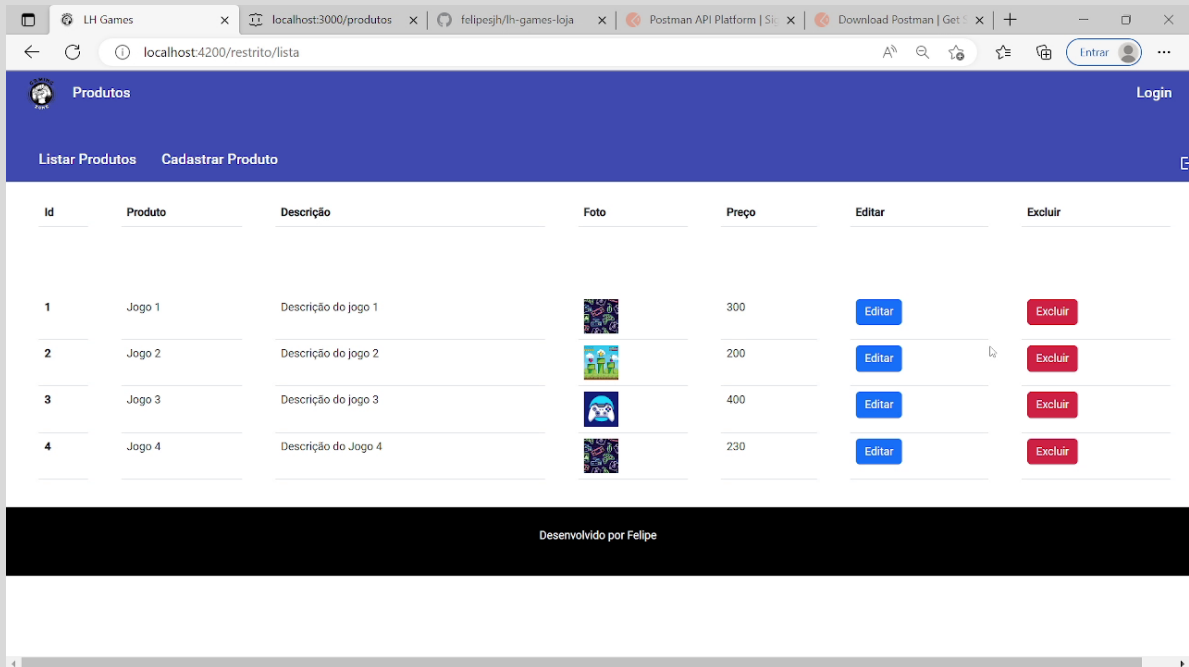
Você sabia?

Foi inserido um comando do Angular chamado **currency**. Ele permite a formatação de moeda.



```
<td>{{p.preco | currency : 'BRL'}}</td>
```

2. Salve os arquivos e faça o teste no navegador. O botão **Excluir** já estará funcionando.



Funcionalidade cadastrar

1. Copie os códigos a seguir nos arquivos indicados da pasta `src > app > restrito > cadastro-produto`.

cadastro-produto.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Produto } from 'src/app/models/Produto.model';
import { ProdutoService } from 'src/app/produto.service';

@Component({
  selector: 'app-cadastro-produto',
  templateUrl: './cadastro-produto.component.html',
  styleUrls: ['./cadastro-produto.component.css']
})
export class CadastroProdutoComponent implements OnInit{

  public produto: Produto = new Produto(0, "", "", "", 0);

  constructor(private _produtoService: ProdutoService, private
router: Router) { }

  ngOnInit(): void {
  }

  cadastrar(){
    this._produtoService.cadastrarProduto(this.produto).subscribe(
      produto => {
        this.produto = new Produto(0, "", "", "", 0);
        alert("Cadastro Efetuado com Sucesso")
      },
      err => {
        alert("erro ao cadastrar")
      }
    );

    this.router.navigate(["/restrito/lista"]);
  }
}
```

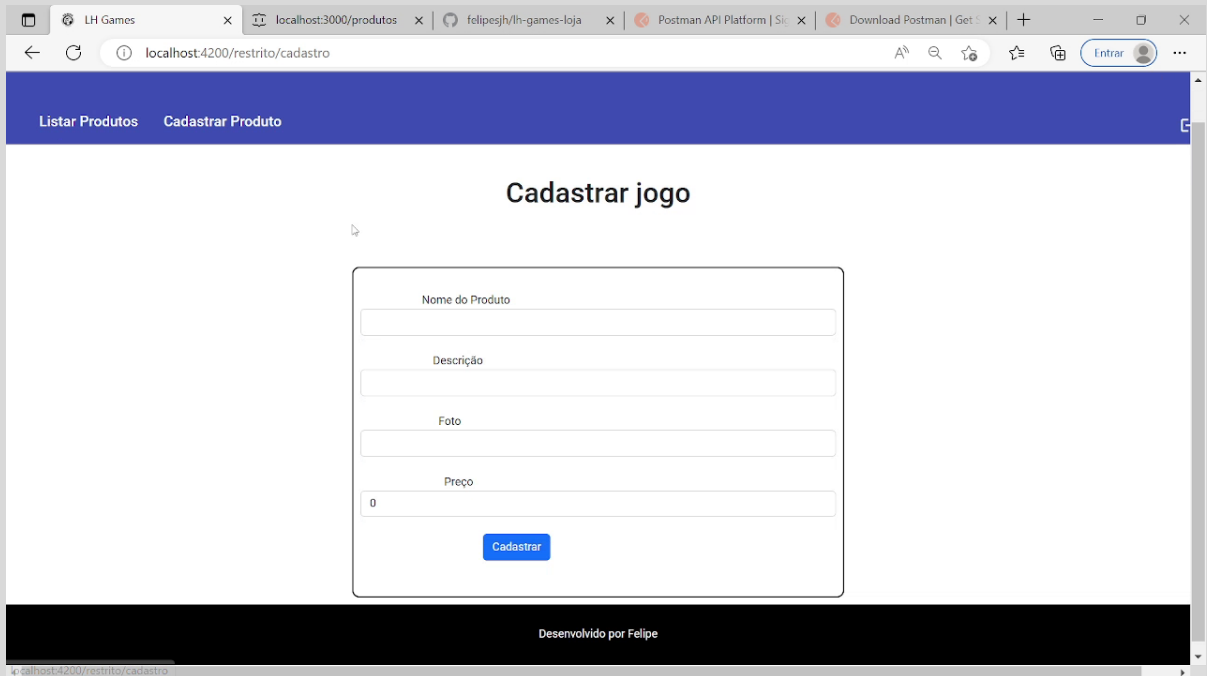
cadastro-produto.component.html

```

<main>
  <h1>Cadastrar jogo</h1>
  <section id="section-cadastro">
    <form>
      <div class="col-auto">
        <label for="produto" >Nome do Produto</label>
        <input type="text" class="form-control" name="produto"
[(ngModel)] = "produto.produto">
      </div>
      <br>
      <div class="col-auto">
        <label for="descricao" >Descrição</label>
        <input type="text" class="form-control"
name="descricao" [(ngModel)] = "produto.descricao">
      </div>
      <br>
      <div class="col-auto">
        <label for="foto" >Foto</label>
        <input type="text" class="form-control" name="foto"
[(ngModel)] = "produto.foto">
      </div>
      <br>
      <div class="col-auto">
        <label for="preco" >Preço</label>
        <input type="number" class="form-control"
name="preco" [(ngModel)] = "produto.preco">
      </div>
      <br>
      <div class="col-auto">
        <button type="button" class="btn btn-primary mb-3"
(click)="cadastrar()">Cadastrar</button>
      </div>
    </form>
  </section>
</main>

```

2. Salve os arquivos e faça o teste no navegador. Agora, aparecerá **Cadastrar Produto** no menu superior da página. Ao clicar nele, é possível preencher um cadastro novo e, após seu preenchimento, ele será gravado e exibido em **Listar Produtos**.



Funcionalidade Atualizar Produto

1. Copie os códigos a seguir nos arquivos indicados da pasta `src > app > restrito > atualiza-produto`.

atualiza-produto.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Produto } from 'src/app/models/Produto.model';
import { ProdutoService } from 'src/app/produto.service';
import { ActivatedRoute, Router } from '@angular/router';
import { map, tap } from 'rxjs';

@Component({
  selector: 'app-atualiza-produto',
  templateUrl: './atualiza-produto.component.html',
  styleUrls: ['./atualiza-produto.component.css']
})
export class AtualizaProdutoComponent implements OnInit {

  public produtoId:number = 0;
  public produto: Produto = new Produto(0,"","","",0);

  constructor(private _produtoService: ProdutoService,private _activatedRoute:
  ActivatedRoute, private _router:Router) {
    this._activatedRoute.params.subscribe(params => this.produtoId =
    params['id']);
  }

  ngOnInit():void{
    this.listarProduto();
  }

  listarProduto():void{
    this._produtoService.getProduto(this.produtoId)
    .subscribe((res:any) => { console.log(res[0].produto);
      this.produto = new
  Produto(res[0].id,res[0].produto,res[0].descricao,res[0].foto,res[0].preco);

    })
  }

  atualizar(id: number){
    this._produtoService.atualizarProduto(id,this.produto).subscribe(
      produto => {this.produto = new Produto(0,"","","",0)},
      err => {console.log("erro ao atualizar")}
    );

    this._router.navigate(["/restrito/lista"]);
  }
}
```

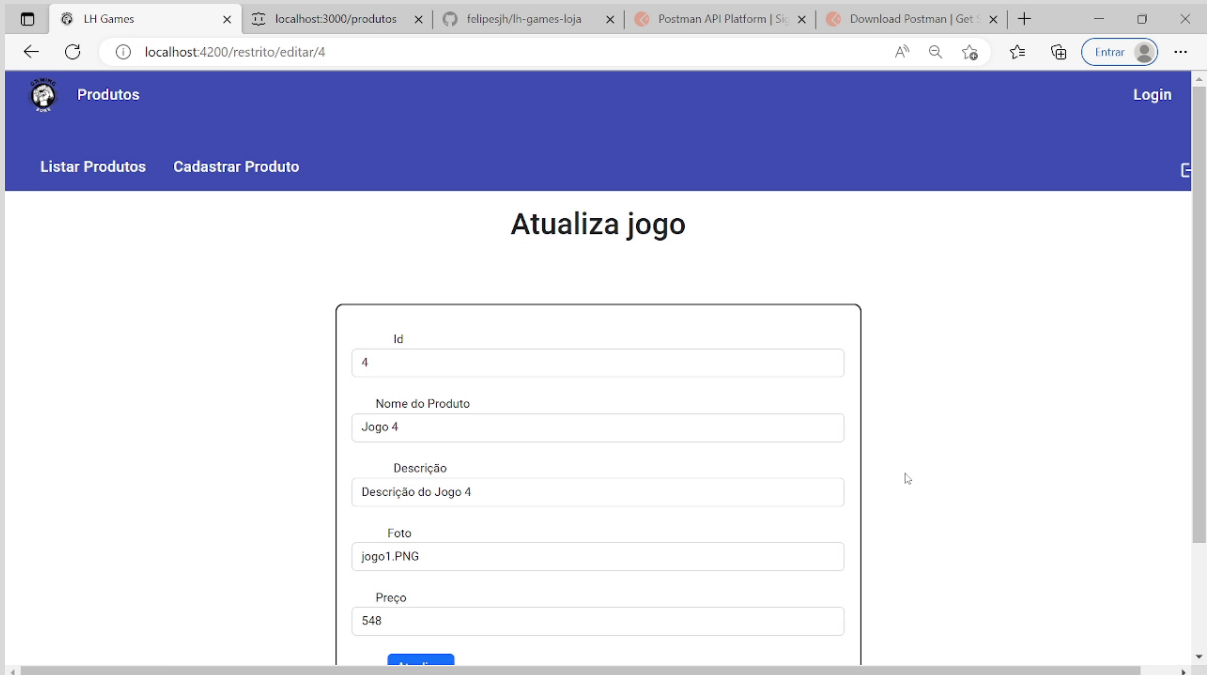
atualiza-produto.component.html

```
<main>

  <h1>Atualiza jogo</h1>
  <section id="section-cadastro">
    <form *ngIf="produto">
      <div class="col-auto">
        <label for="produto">Id</label>
        <input type="text" class="form-control" name="id"
[(ngModel)]="produto.id" readonly>
      </div>
      <br>
      <div class="col-auto">
        <label for="produto">Nome do Produto</label>
        <input type="text" class="form-control" name="produto"
[(ngModel)]="produto.produto" >
      </div>
      <br>
      <div class="col-auto">
        <label for="descricao">Descrição</label>
        <input type="text" class="form-control" name="descricao"
[(ngModel)]="produto.descricao" >
      </div>
      <br>
      <div class="col-auto">
        <label for="foto">Foto</label>
        <input type="text" class="form-control" name="foto"
[(ngModel)]="produto.foto">
      </div>
      <br>
      <div class="col-auto">
        <label for="preco">Preço</label>
        <input type="number" class="form-control" name="preco"
[(ngModel)]="produto.preco" >
      </div>
      <br>
      <div class="col-auto">
        <button type="submit" class="btn btn-primary mb-3"
(click)="atualizar(produto.id)">Atualizar</button>
      </div>
    </form>
  </section>

</main>
```

2. Salve os arquivos e verifique no navegador o resultado. Ao clicar em **Editar**, é possível modificar os valores de um produto listado.



The screenshot shows a web browser window with the URL `localhost:4200/restrito/editar/4`. The page has a blue header with the title 'Produtos' and a 'Login' button. Below the header, there are two links: 'Listar Produtos' and 'Cadastrar Produto'. The main content area is titled 'Atualiza jogo' and contains a form with the following fields:

- Id**: A text input field containing the value '4'.
- Nome do Produto**: A text input field containing the value 'Jogo 4'.
- Descrição**: A text input field containing the value 'Descrição do Jogo 4'.
- Foto**: A text input field containing the value 'jogo1.PNG'.
- Preço**: A text input field containing the value '548'.

At the bottom of the form, there is a blue button labeled 'Atualizar'.

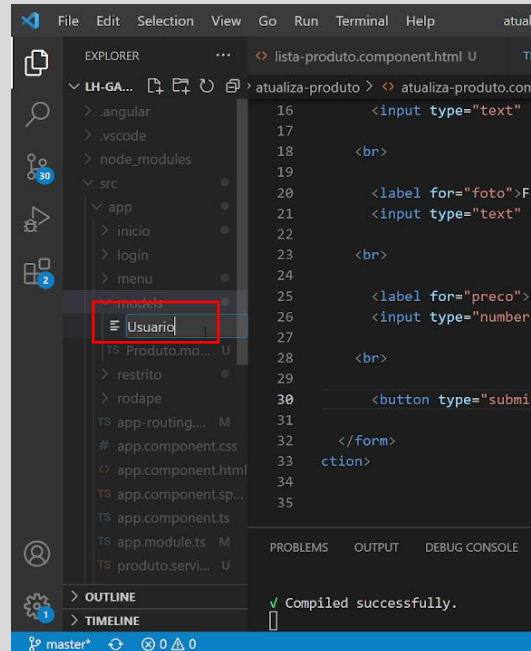
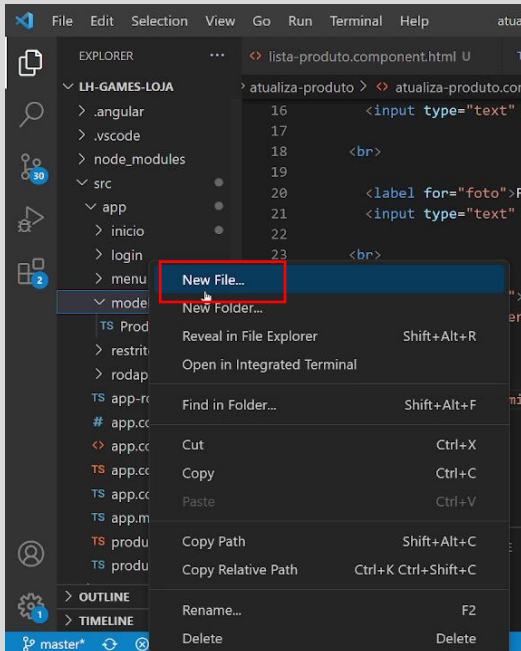
Dica!

Em `atualiza-produto.component.html`, inserimos **readonly** na **Id** do produto para que não seja possível modificá-lo.



Sistema de login

1. No VSCode, localize a pasta `models`, clique com o botão direito do mouse e selecione `New File`. Nomeie esse arquivo como **Usuario.model.ts**.



2. Abra o arquivo criado **Usuario.model.ts** e altere o código conforme abaixo.

```
export class Usuario{  
  
    public login?: string;  
    public senha?: string;  
  
}
```

3. No Terminal, selecione o cmd à direita e digite o comando abaixo para gerar o arquivo de login.

```
ng g s login
```

4. Agora, abra o arquivo que acabou de criar **login.service.ts**, localizado na pasta app, e substitua o código a seguir.

```
import { HttpClient } from '@angular/common/http';
import { EventEmitter, Injectable } from '@angular/core';
import { BehaviorSubject, Observable, of, Subject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class LoginService {

  mostraMenu = new Subject<boolean>()

  constructor() { }

  login(usuario:string, senha:string){
    if(usuario=="aluno" && senha=="1234"){
      localStorage.setItem('token','qwer1234');
      this.mostraMenu.next(false)
    }else{
      this.mostraMenu.next(true);
      window.location.reload();
    }
  }

  setMostraMenu(valor: boolean) {
    this.mostraMenu.next(valor)
  }

  getMostraMenu() {
    return this.mostraMenu.asObservable();
  }
}
```


Importante!

Para a área de login, além de nome de usuário e senha, também, é necessário criar um token que libere o acesso do usuário às áreas restritas do site.

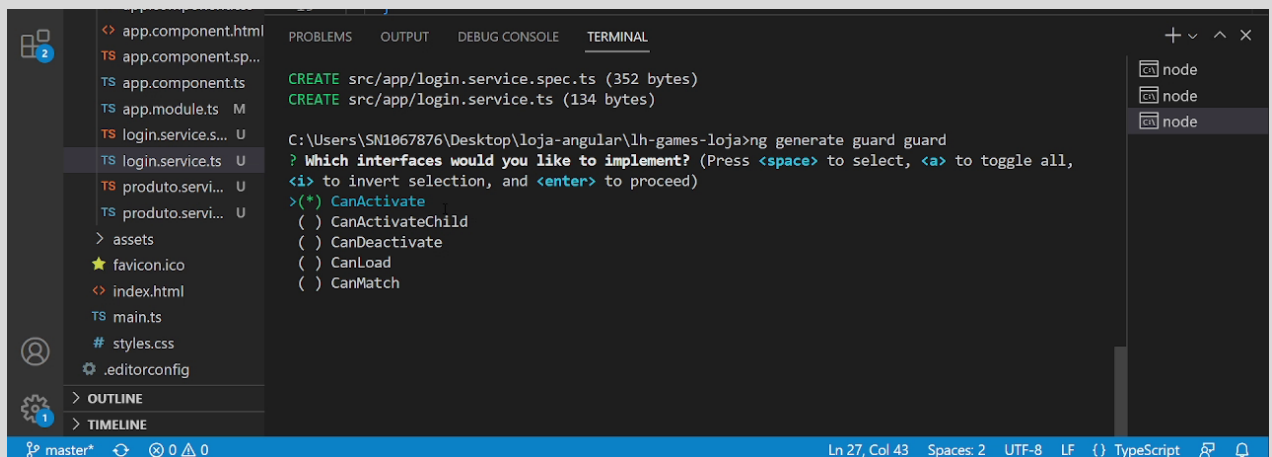


Arquivo guard (restrição)

1. No Terminal, selecione o cmd à direita e digite o comando abaixo para gerar o arquivo guard.

```
ng generate guard guard
```

2. O Terminal exibirá uma pergunta. Pressione **Enter** para aceitar a primeira opção (CanActivate).



3. Abra o arquivo criado **guard.guard.ts**. Substitua o conteúdo do código por este que aparece abaixo.

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, CanActivate, RouterStateSnapshot,
  UrlTree } from '@angular/router';
import { Observable } from 'rxjs';
import { Router } from '@angular/router';

@Injectable({
  providedIn: 'root'
})
export class GuardGuard implements CanActivate {

  constructor(private router: Router){}

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> |
  Promise<boolean | UrlTree> | boolean | UrlTree {

    if(localStorage.getItem('token') !== null) {
      return true;
    }

    this.router.navigate(['/login'])
    return false;
  }
}
```

4. Abra novamente o arquivo **app-routing.module.ts** para adicionar Guard nas rotas. Substitua a linha de código destacada abaixo para inserir o CanActivate nela.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { InicioComponent } from './inicio/inicio.component';
import { LoginComponent } from './login/login.component';
import { RestritoComponent } from './restrito/restrito.component';
import { GuardGuard } from './guard.guard';

const routes: Routes = [
  {path: 'inicio', component: InicioComponent},
  {path: 'login', component: LoginComponent},
  {path: 'restrito', component: RestritoComponent, canActivate: [GuardGuard]},
  {path: '', redirectTo: '/inicio', pathMatch: 'full'}
]

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

5. Também, é necessário adicionar o CanActivate no arquivo **restrito-routing.module.ts**, conforme destacado abaixo.

```
import {NgModule} from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { CadastroProdutoComponent } from '../cadastro-produto/cadastro-produto.component';
import { ListaProdutoComponent } from '../lista-produto/lista-produto.component';
import { AtualizaProdutoComponent } from '../atualiza-produto/atualiza-produto.component';
import { RestritoComponent } from '../restrito.component';
import { GuardGuard } from '../guard.guard';

const restritoRoutes: Routes = [
  {path: 'restrito', component: RestritoComponent, children:[
    {path: 'cadastro', component: CadastroProdutoComponent, canActivate:
[GuardGuard]},
    {path: 'lista', component: ListaProdutoComponent, canActivate:
[GuardGuard]},
    {path: 'editar/:id', component: AtualizaProdutoComponent, canActivate:
[GuardGuard]}
  ]},
  {path: '', redirectTo: '/restrito/lista', pathMatch:'full'}
]

@NgModule({
  imports: [RouterModule.forChild(restritoRoutes)],
  exports: [RouterModule]
})

export class RestritoRoutingModule{
}
```

6. Em **app.component.html**, faça a substituição pelo código a seguir.

```
<app-menu *ngIf="mostrarMenu"></app-menu>
<router-outlet></router-outlet>
```

7. Em **app.component.ts**, substitua pelo código a seguir.

```
import { Component, HostListener, OnDestroy, OnInit } from
 '@angular/core';
import { LoginService } from '../login.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit, OnDestroy {
  title = 'LH Games';
  mostrarMenu:boolean = true;

  constructor(private _loginService: LoginService){}

  ngOnInit(){

    this._loginService.getMostraMenu().subscribe(res => {
      this.mostrarMenu = res;
    })
  }

  ngOnDestroy() {
    localStorage.clear();
  }
}
```

8. Agora, localize o **login.components.ts** para criar um usuário nele. Copie o código abaixo.

```
import { Component, OnInit } from '@angular/core';
import { LoginService } from '../login.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit{

  usuario!: string;
  senha!:string;

  constructor(private _loginService:LoginService,private _router:
Router) { }

  ngOnInit():void{

  }

  fazerLogin(){

    this._loginService.login(this.usuario,this.senha);
    this._router.navigate(['/restrito/lista']);

    this._loginService.setMostraMenu(false)

  }
}
```

9. Abra o **login.component.html** para realizar as alterações nesse código, conforme indicado abaixo.

```
<main>
  <section id="section-login">
    <form>
      <div>
        <label for="usuario">Email</label>
        <input name="usuario" required type="text"
[(ngModel)]="usuario">
        <br>
        <label for="password">Password</label>
        <input name="password" type="password"
[(ngModel)]="senha">
      </div>
      <button (click)="fazerLogin()">Login</button>
    </form>
  </section>
</main>
```

10. Altere o arquivo `lista-produto.component.ts` conforme abaixo.

```

import { Component, HostListener } from '@angular/core';
import { Router } from '@angular/router';
import { LoginService } from 'src/app/login.service';
import { Produto } from 'src/app/models/Produtos.model';
import { ProdutoService } from 'src/app/produto.service';

@Component({
  selector: 'app-lista-produto',
  templateUrl: './lista-produto.component.html',
  styleUrls: ['./lista-produto.component.css']
})

export class ListaProdutoComponent {
  public produtos: Produto[] = [ ];
  public produto: Produto = new Produto(0, "", "", "", 0);

  constructor(private _produtoService: ProdutoService,
    private router: Router,
    private _loginService: LoginService){}

  ngOnInit():void{
    this.listarProdutos();
    this._loginService.setMostraMenu(false);
  }

  listarProdutos():void{
    this._produtoService.getProdutos()
      .subscribe(
        retornaProduto => {
          this.produtos = retornaProduto.map(
            item => {
              return new Produto(
                item.id,
                item.produto,
                item.descricao,
                item.foto,
                item.preco
              );
            }
          );
        }
      )
  }

  excluir(id: number){
    this._produtoService.removerProduto(id).subscribe(
      vaga => {
        this.listarProdutos();
      },
      err => {console.log("erro ao Excluir")}
    );

    // window.location.href = "/restrito/lista";
    this.router.navigate(["/restrito/lista"]);
  }
}

```


Funcionalidade logout

1. Altere o arquivo **menu-restrito.component.ts** conforme abaixo.

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { LoginService } from 'src/app/login.service';

@Component({
  selector: 'app-menu-restrito',
  templateUrl: './menu-restrito.component.html',
  styleUrls: ['./menu-restrito.component.css']
})
export class MenuRestritoComponent {

  constructor(private router: Router,
    private _loginService: LoginService){}

  logout(){
    localStorage.clear();
    this._loginService.setMostraMenu(true);
    this.router.navigate(['/login']);
  }
}
```

2. Abra o arquivo **menu-restrito.components.html** e realize a alteração abaixo.

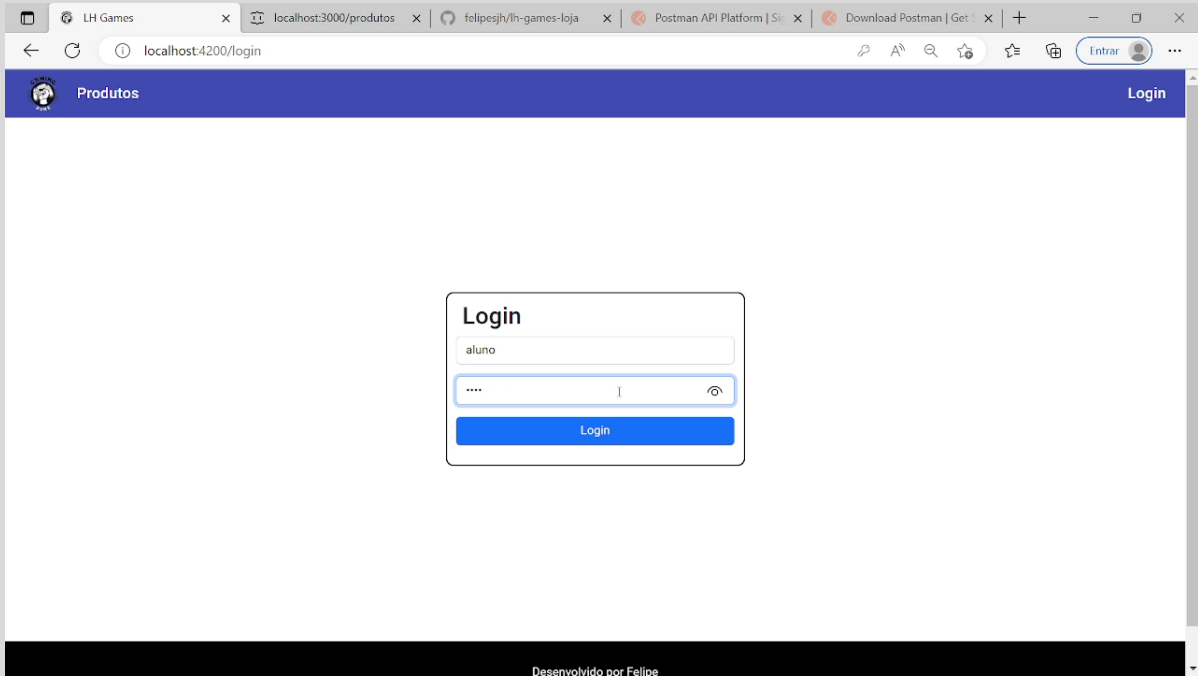
```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { LoginService } from 'src/app/login.service';

@Component({
  selector: 'app-menu-restrito',
  templateUrl: './menu-restrito.component.html',
  styleUrls: ['./menu-restrito.component.css']
})
export class MenuRestritoComponent {

  constructor(private router: Router,
    private _loginService: LoginService){}

  logout(){
    localStorage.clear();
    this._loginService.setMostraMenu(true);
    this.router.navigate(['/login']);
  }
}
```

3. Salve os arquivos e teste no navegador as funcionalidades login e logout.



Você sabia?

Se um usuário deslogado tentar abrir uma das páginas restritas, por exemplo, **/restrito/lista**, ele será redirecionado para a página de login. O usuário só consegue visualizar as páginas restritas se obtiver o token de acesso.

