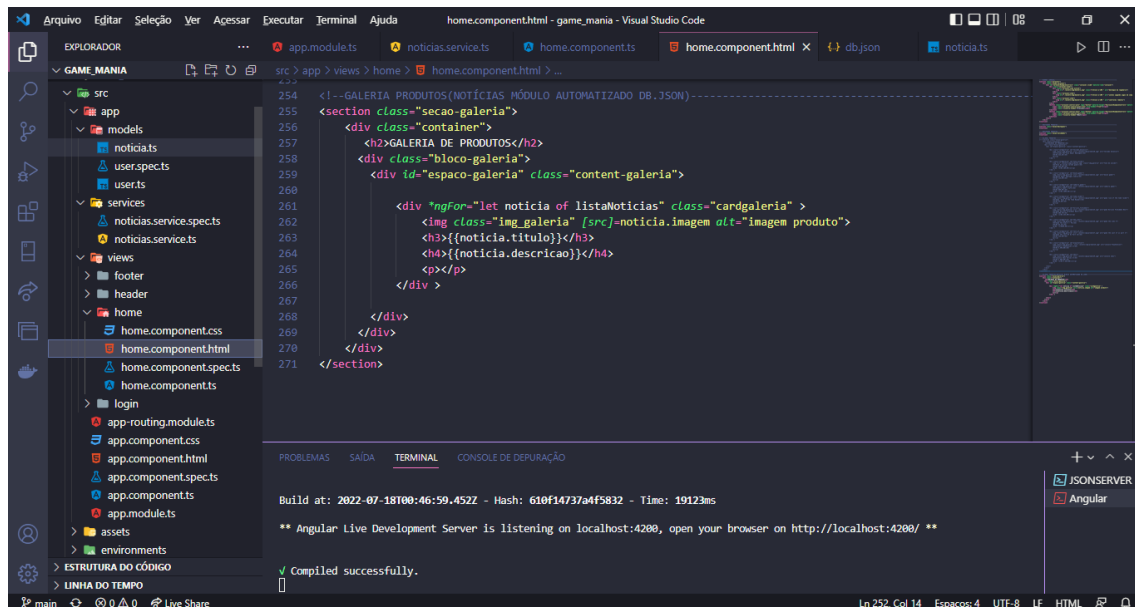


ATIVIDADES E ENCONTRO REMOTO 1 – HOME

➔ Automatizando exibição de produtos com o db.json (GET)



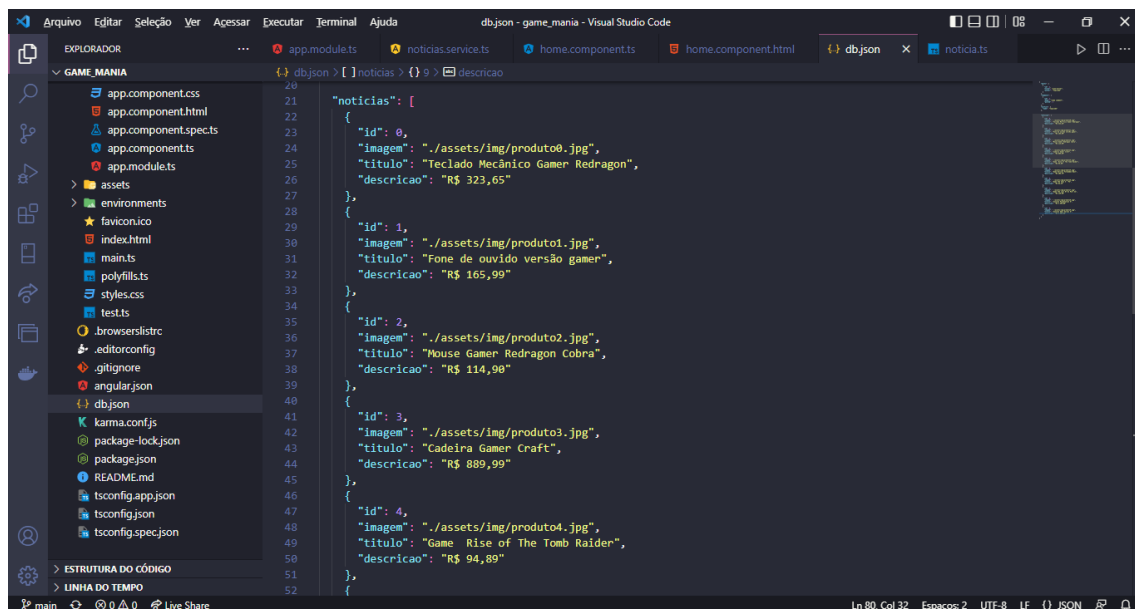
The screenshot shows the Visual Studio Code interface with the `home.component.html` file open. The file contains HTML code for a product gallery. The terminal at the bottom shows the output of the Angular Live Development Server, indicating that the application is running successfully on `localhost:4200`.

```
<!-- GALERIA PRODUTOS (NOTÍCIAS MÓDULO AUTOMATIZADO DB.JSON) -->
<section class="secao-galeria">
  <div class="container">
    <h2>GALERIA DE PRODUTOS</h2>
    <div class="bloco-galeria">
      <div id="espaco-galeria" class="content-galeria">
        <div *ngFor="let noticia of listaNoticias" class="cardgaleria">
          <img class="img_galeria" [src]="noticia.imagem" alt="imagem produto">
          <h3>{{noticia.titulo}}</h3>
          <h4>{{noticia.descricao}}</h4>
        </div>
      </div>
    </div>
  </div>
</section>
```

Build at: 2022-07-18T00:46:59.452Z - Hash: 610f14737a4f5832 - Time: 19123ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.



The screenshot shows the Visual Studio Code interface with the `db.json` file open. The file contains a JSON array of product data.

```
{
  "noticias": [
    {
      "id": 0,
      "imagem": "./assets/img/produto0.jpg",
      "titulo": "Teclado Mecânico Gamer Redragon",
      "descricao": "R$ 323,65"
    },
    {
      "id": 1,
      "imagem": "./assets/img/produto1.jpg",
      "titulo": "Fone de ouvido versão gamer",
      "descricao": "R$ 165,99"
    },
    {
      "id": 2,
      "imagem": "./assets/img/produto2.jpg",
      "titulo": "Mouse Gamer Redragon Cobra",
      "descricao": "R$ 114,90"
    },
    {
      "id": 3,
      "imagem": "./assets/img/produto3.jpg",
      "titulo": "Cadeira Gamer Craft",
      "descricao": "R$ 889,99"
    },
    {
      "id": 4,
      "imagem": "./assets/img/produto4.jpg",
      "titulo": "Game Rise of The Tomb Raider",
      "descricao": "R$ 94,89"
    }
  ]
}
```

```
src > app > services > noticias.service.ts > NoticiasService > getNoticias
1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { Observable } from 'rxjs';
4 import { Noticia } from '../models/noticia';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class NoticiasService {
10
11   url = "http://localhost:3000/noticias"
12
13   constructor(private httpClient: HttpClient) { }
14
15   getNoticias(): Observable<Noticia[]> {
16     return this.httpClient.get<Noticia[]>(this.url)
17   }
18 }
19
```

```
src > app > views > home > home.component.ts > HomeComponent > ngOnInit
1 import { Component, OnInit } from '@angular/core';
2 import { OwlOptions } from 'ngx-owl-carousel-o';
3 import { Noticia } from 'src/app/models/noticia';
4 import { NoticiasService } from 'src/app/services/noticias.service';
5
6 @Component({
7   selector: 'app-home',
8   templateUrl: './home.component.html',
9   styleUrls: ['./home.component.css']
10 })
11 export class HomeComponent implements OnInit {
12
13   constructor(private noticiaService: NoticiasService) { }
14
15   listaNoticias = [] as Noticia[]
16
17   ngOnInit(): void {
18     this.carregarNoticias()
19   }
20
21   carregarNoticias(){
22     this.noticiaService.getNoticias().subscribe((noticiasRecebidas: Noticia[])=>{
23       this.listaNoticias = noticiasRecebidas;
24       console.log(this.listaNoticias);
25     })
26   }
27
28   customOptions: OwlOptions = {
29     loop: false,
30     mouseDrag: true,
31     touchDrag: true,
32     pullDrag: true,
33     date: false
34   }
35 }
36
```

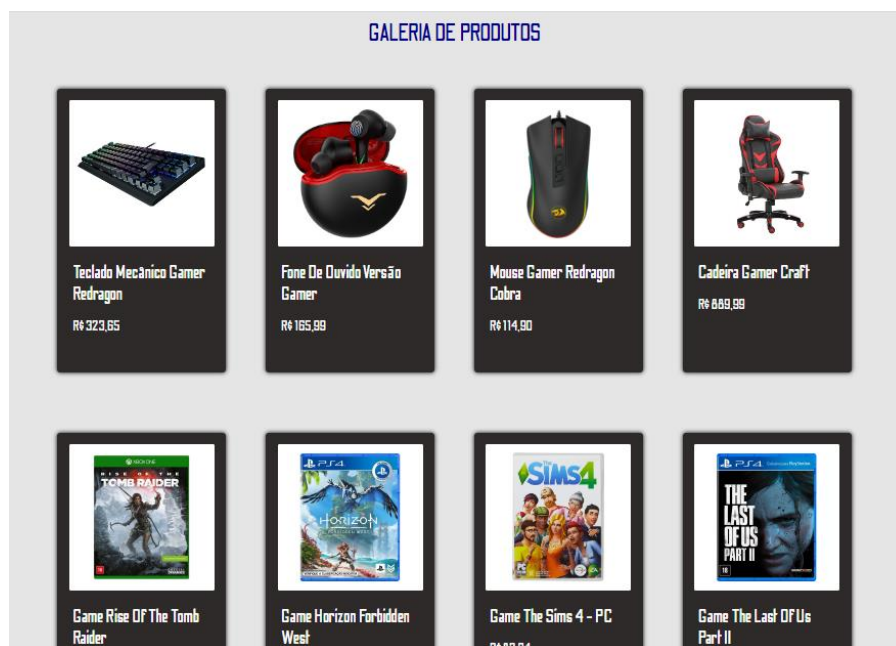
```
src > app > models > noticia.ts > Noticia > imagem
1 export interface Noticia {
2   id?: number;
3   titulo: string;
4   descricao: string;
5   imagem: string;
6 }
7
```

```
Arquivo  Editar  Seleção  Ver  Acessar  Executar  Terminal  Ajuda  • app.module.ts - game_mania - Visual Studio Code

EXPLORADOR  src > app > app.module.ts > AppModule
  GAME MANIA
  app
  models
  noticias
  user.spects
  users
  services
  noticias.service.spects
  noticias.service.ts
  views
  footer
  header
  home
  home.component.css
  home.component.html
  home.component.spects
  home.component.ts
  login
  app-routing.module.ts
  app.component.css
  app.component.html
  app.component.spects
  app.component.ts
  app.module.ts
  assets
  environments
  faviconico
  Estrutura do Código
  Linha do Tempo

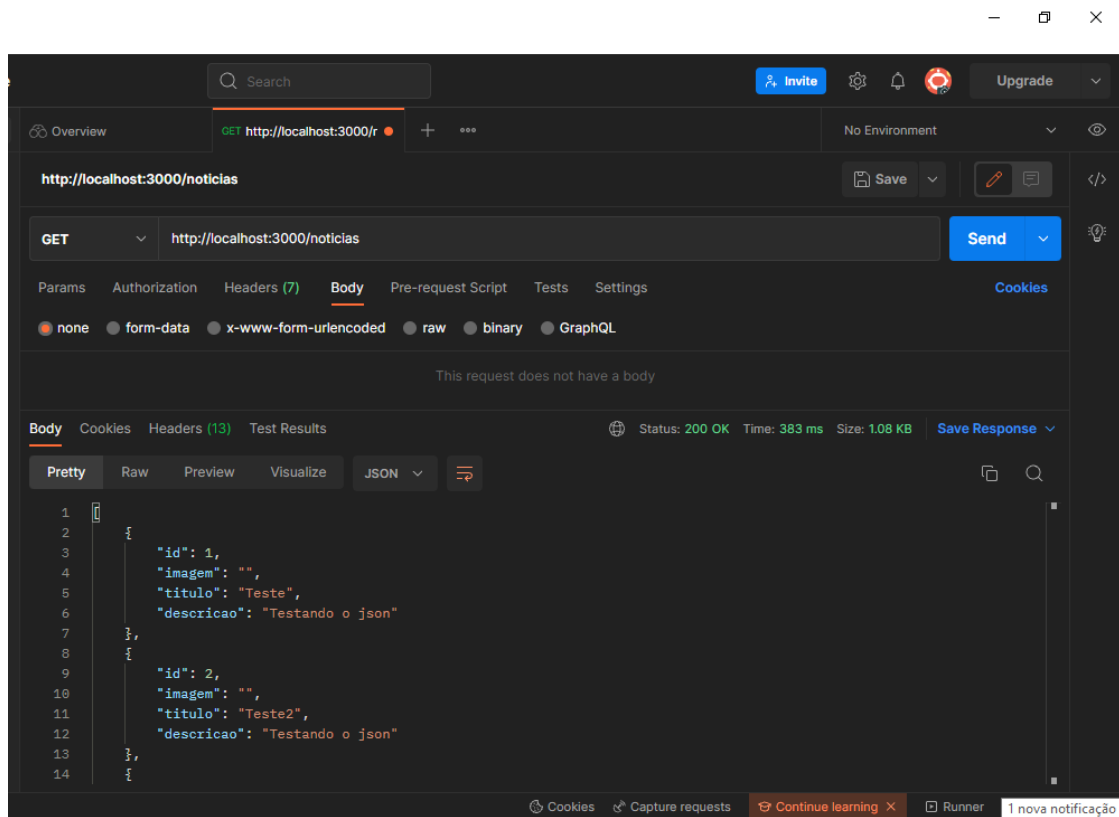
10  import { FooterComponent } from './views/footer/footer.component';
11  import { LoginComponent } from './views/login/login.component';
12
13  import { FormsModule } from '@angular/forms'; /*formulario*/
14  import { MatButtonModule } from '@angular/material/button'; /*botao formulario*/
15
16  import { HttpClientModule } from '@angular/common/http';
17
18  @NgModule({
19    declarations: [
20      AppComponent,
21      HeaderComponent,
22      HomeComponent,
23      FooterComponent,
24      LoginComponent
25    ],
26    imports: [
27      BrowserModule,
28      AppRoutingModule,
29      BrowserModuleAnimationsModule,
30      CarouselModule, /*carrousel*/
31      FormsModule, /*formulario */
32      MatButtonModule, /*botao formulario*/
33      HttpClientModule /*para fazer requisicoes pelo angular e nao pelo postman */
34    ],
35    providers: [],
36    bootstrap: [AppComponent]
37  })
38  export class AppModule { }
39
```

Resultado:

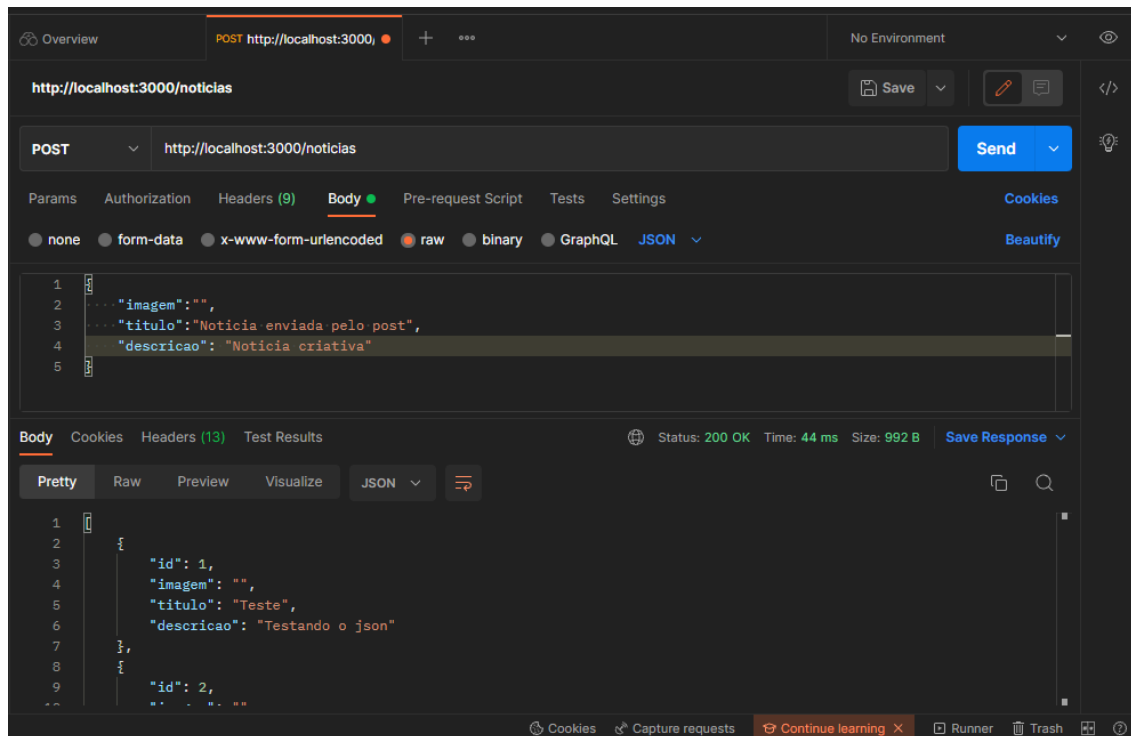


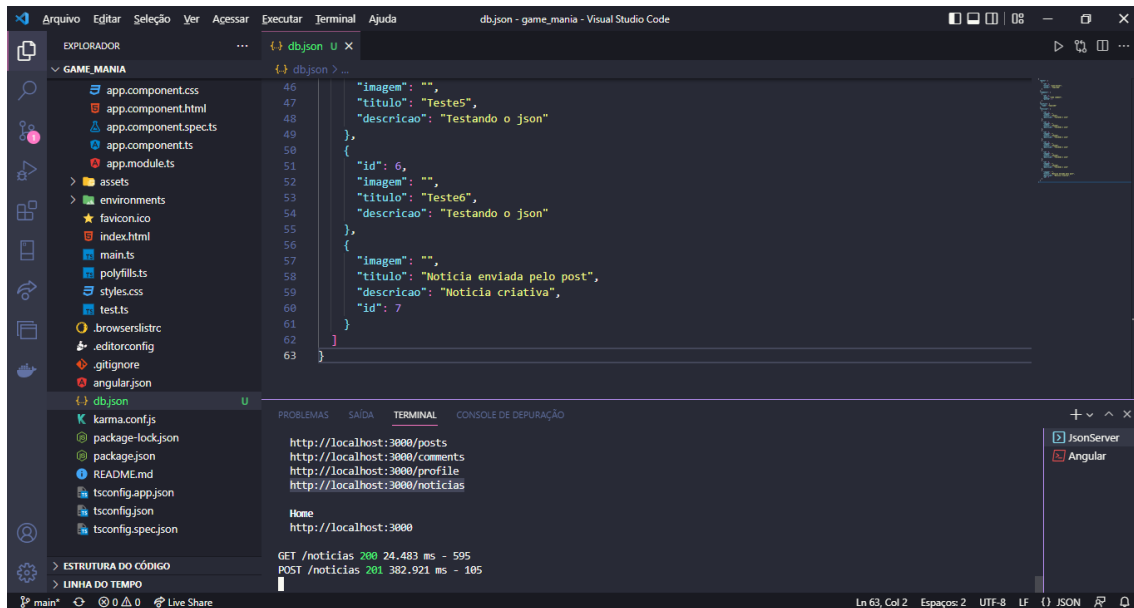
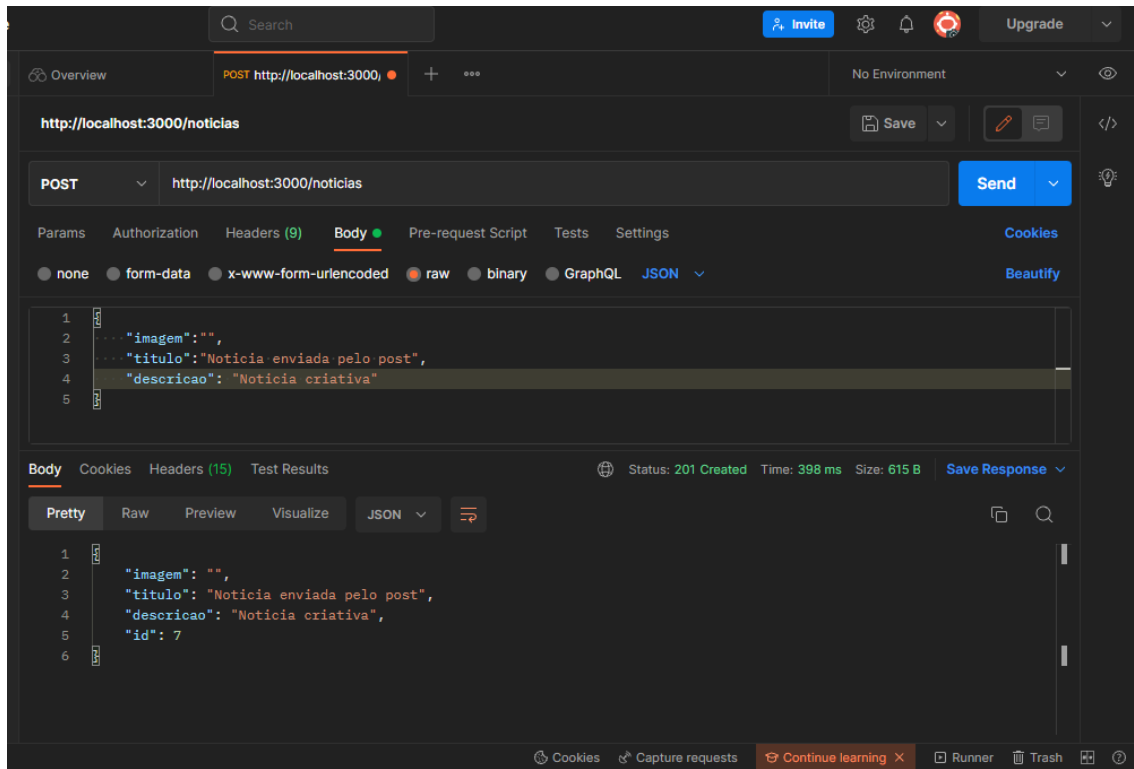
TESTANDO COMANDOS GET, POST, DELETE NO POSTMAN - HOME

➔ Realizando consulta via GET

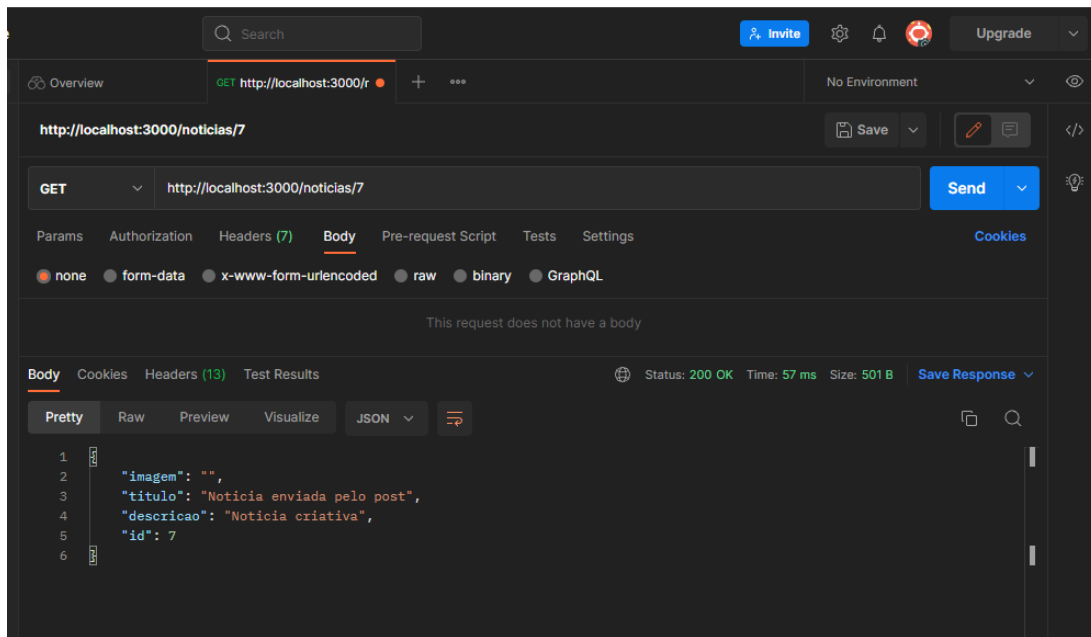


➔ Inserindo dados via POST:

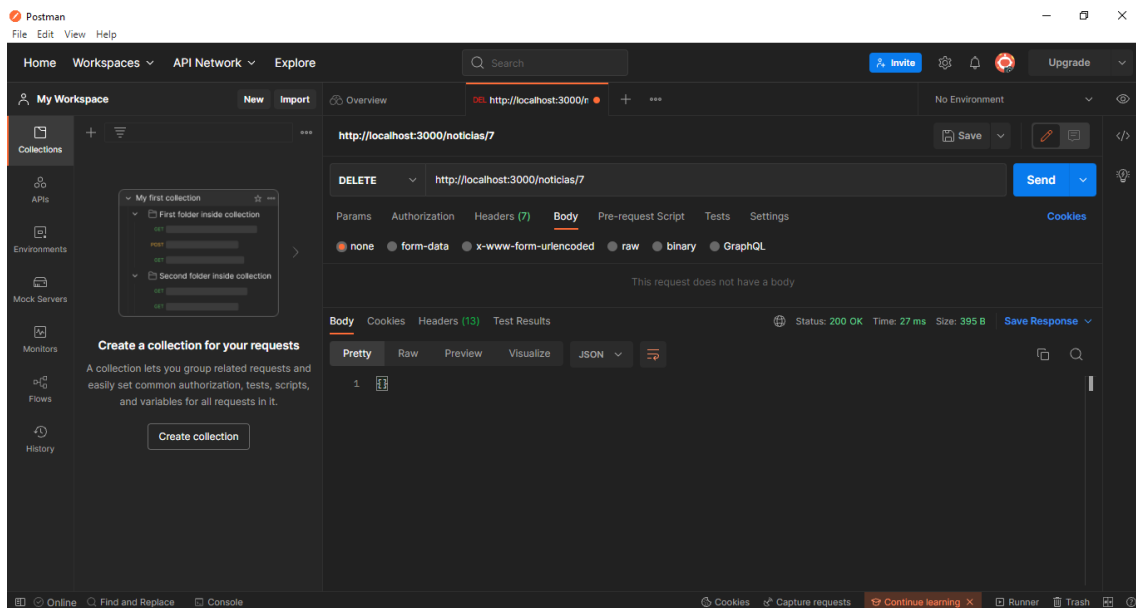


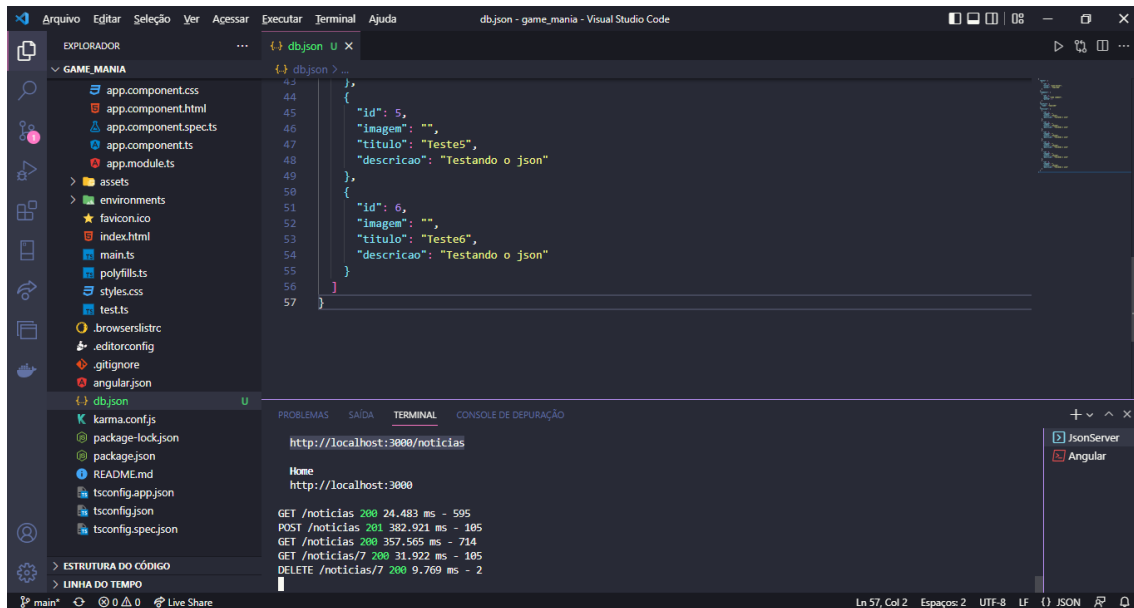


GET do item criado



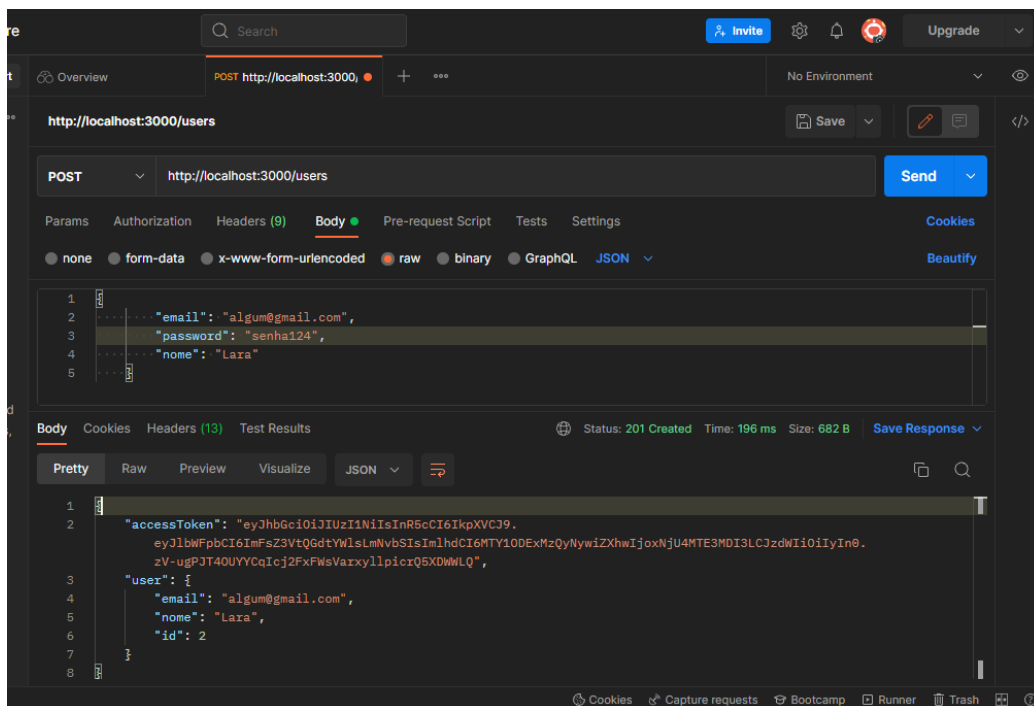
➔ DELETE do item 7

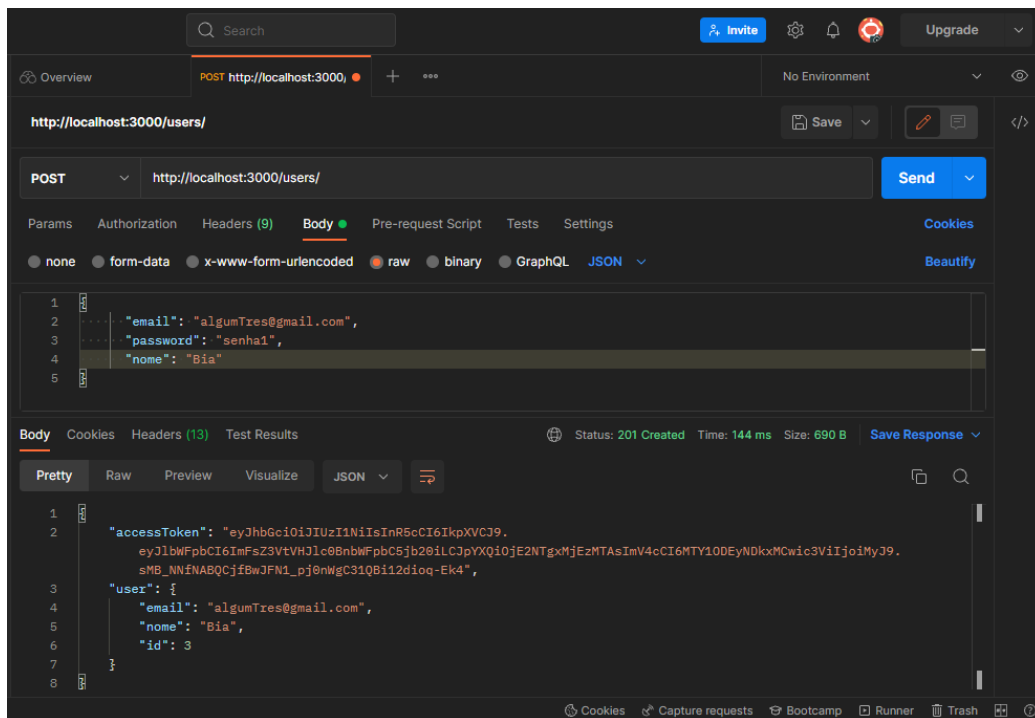




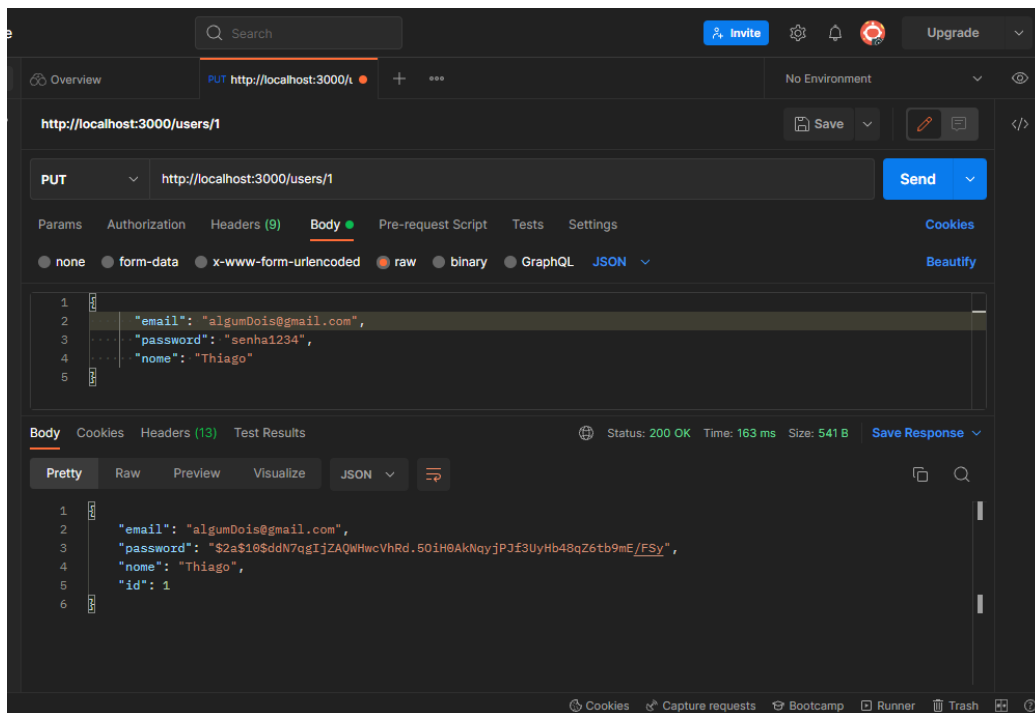
ATIVIDADES E ENCONTRO REMOTO 2 - LOGIN

➔ Criando login com senha criptografada com json-server-auth (POST)





➔ Alterando cadastro com PUT



→ Consultando com GET

The screenshot shows the Postman interface with a GET request to `http://localhost:3000/users/`. The request is successful, returning a 200 OK status. The response body is displayed in JSON format, showing an array of three user objects. The status bar indicates a response time of 72 ms and a size of 867 B.

```
1 {
2   {
3     "email": "algunDois@gmail.com",
4     "password": "$2a$10$dN7qgIjZAQWwVhRd.50iH0AkNqyJP3f3UyHb48qZ6tb9mE/FSy",
5     "nome": "Thiago",
6     "id": 1
7   },
8   {
9     "email": "algun@gmail.com",
10    "password": "$2a$10$/Cj/Zeh07AZ2oun9t.2f4.hI/ev9AXEdfSKAs.gEWX8EV8Yq/Tzbi",
11    "nome": "Lara",
12    "id": 2
13  },
14  {
15    "email": "algunTres@gmail.com",
16    "password": "$2a$10$/2zfRyTETwahCKEEH9jvNuVbD2sq.mIzhjg7kFw7h/BKMld9QRazK",
17    "nome": "Bia",
18    "id": 3
19  }
20 }
```

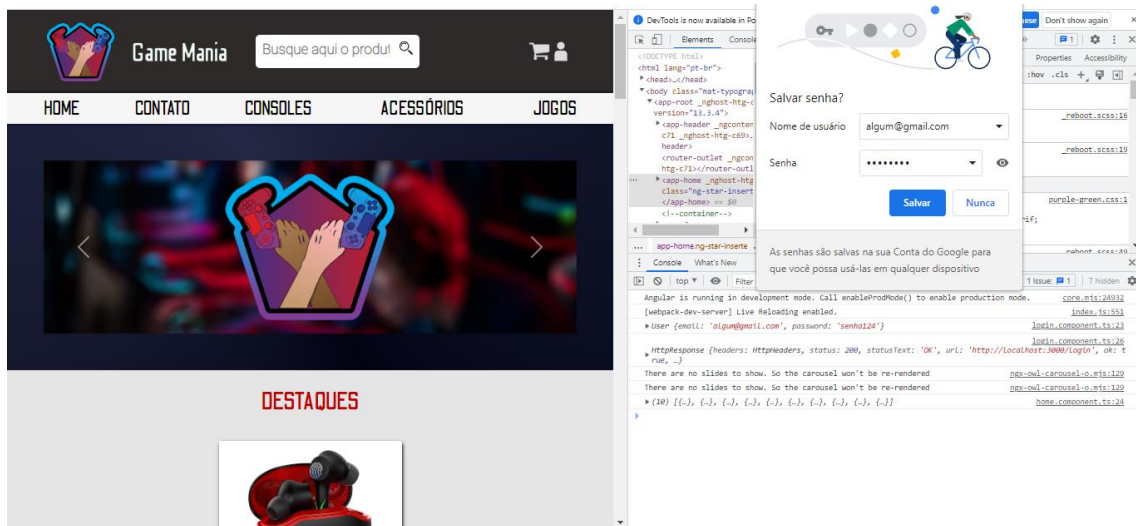
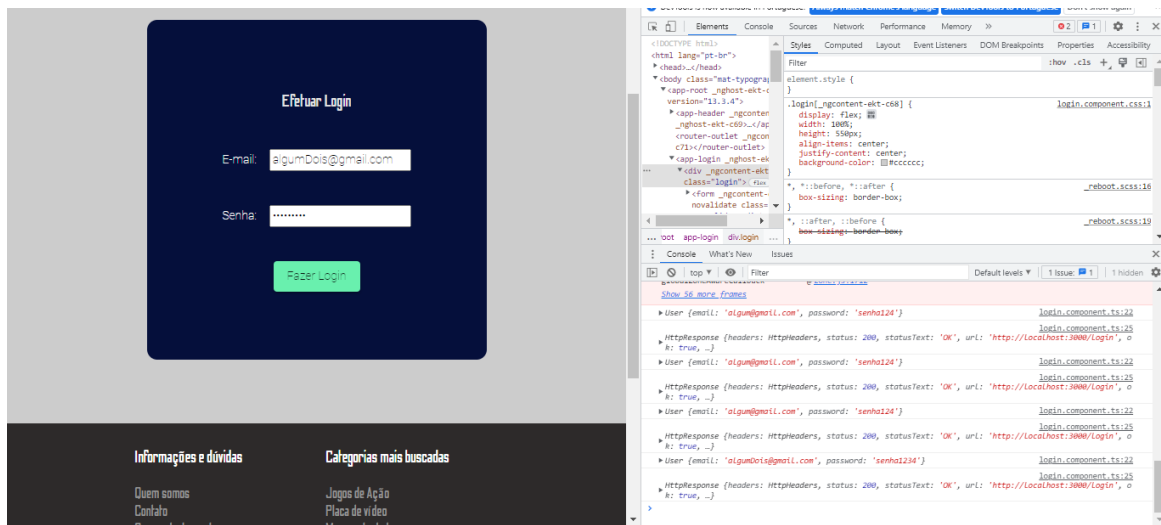
→ Excluindo com DELETE

The screenshot shows the Postman interface with a DELETE request to `http://localhost:3000/users/3`. The request is successful, returning a 200 OK status. The response body is empty, as indicated by the message "This request does not have a body". The status bar indicates a response time of 59 ms and a size of 395 B.

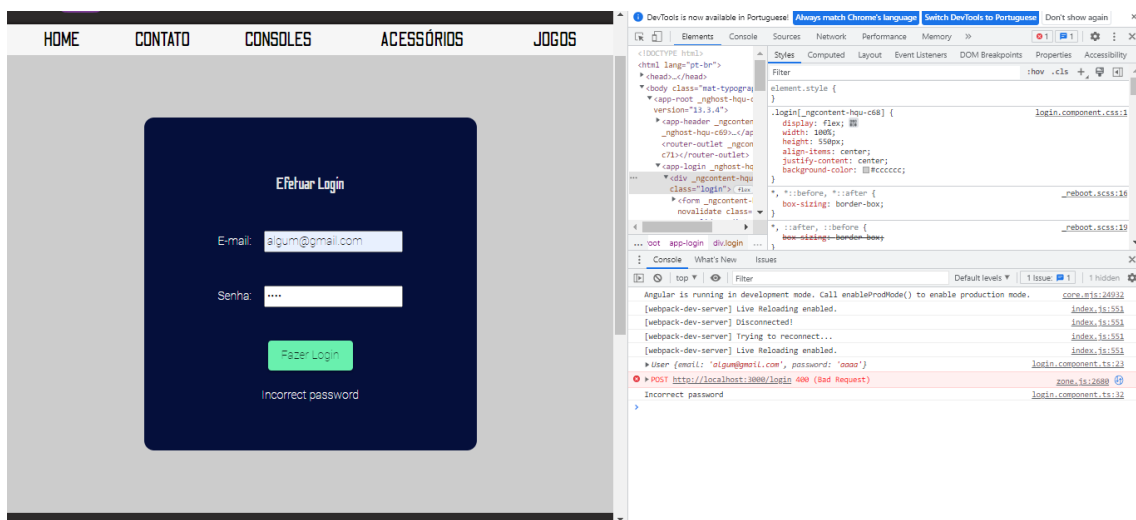
The screenshot shows the Postman interface with a GET request to `http://localhost:3000/users/`. The request is successful, returning a 200 OK status. The response body is displayed in JSON format, showing an array of two user objects. The status bar indicates a response time of 43 ms and a size of 711 B.

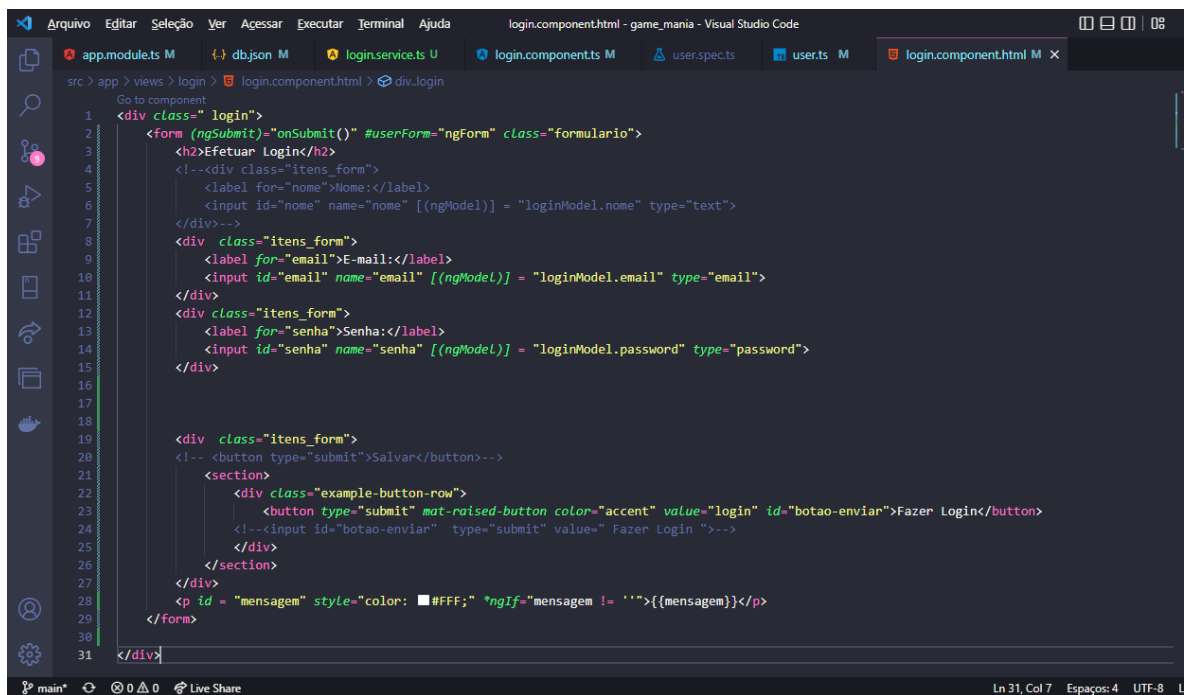
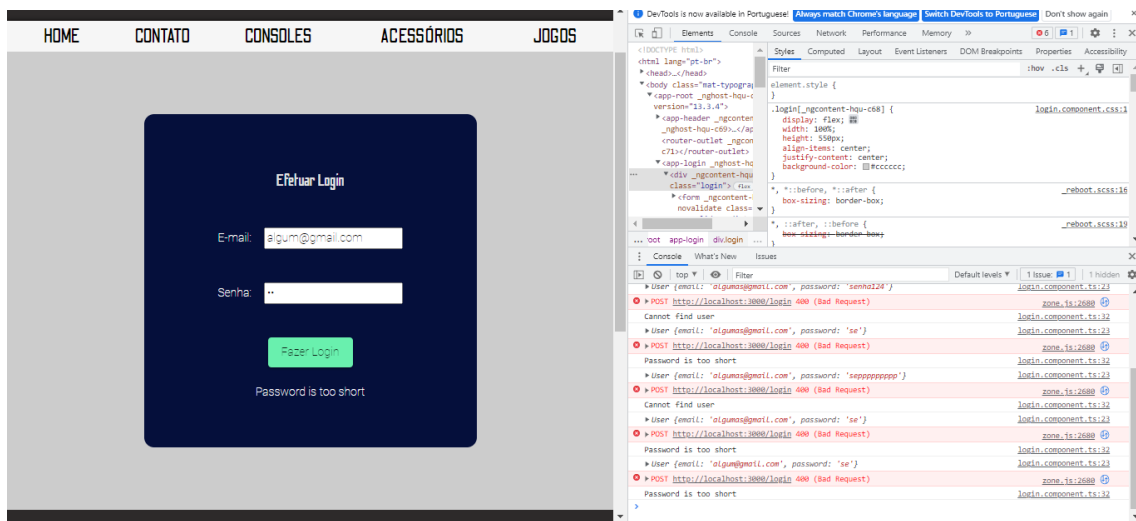
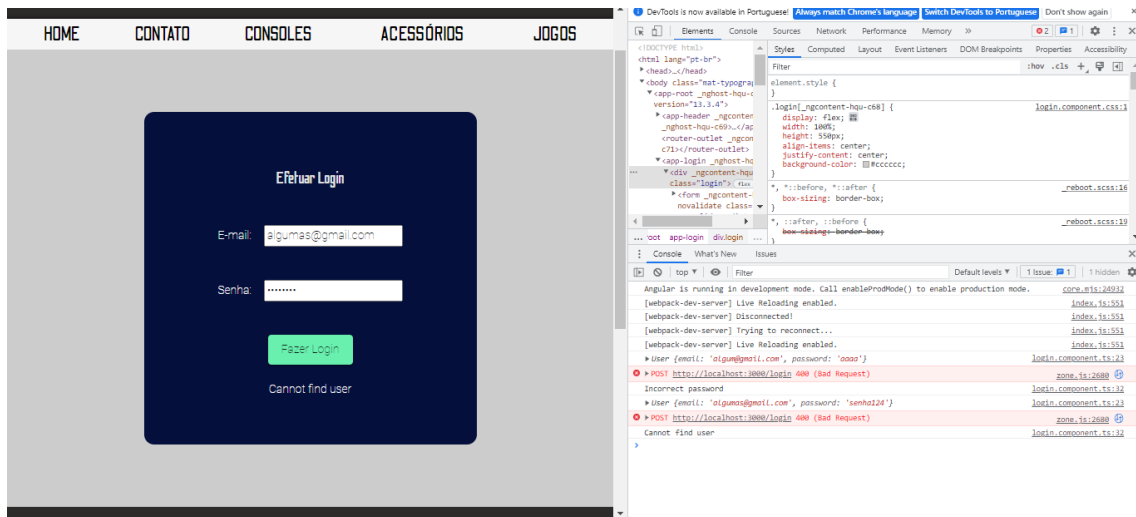
```
1 {
2   {
3     "email": "algunDois@gmail.com",
4     "password": "$2a$10$dN7qgIjZAQWwVhRd.50iH0AkNqyJP3f3UyHb48qZ6tb9mE/FSy",
5     "nome": "Thiago",
6     "id": 1
7   },
8   {
9     "email": "algun@gmail.com",
10    "password": "$2a$10$/Cj/Zeh07AZ2oun9t.2f4.hI/ev9AXEdfSKAs.gEWX8EV8Yq/Tzbi",
11    "nome": "Lara",
12    "id": 2
13  }
14 }
```

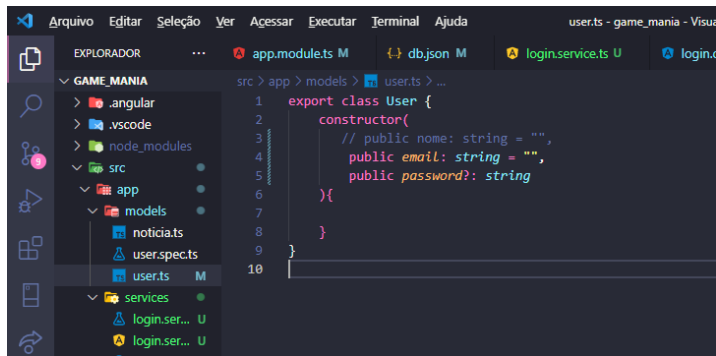
VERIFICANDO E-MAIL E SENHA NO LOGIN COM MENSAGEM



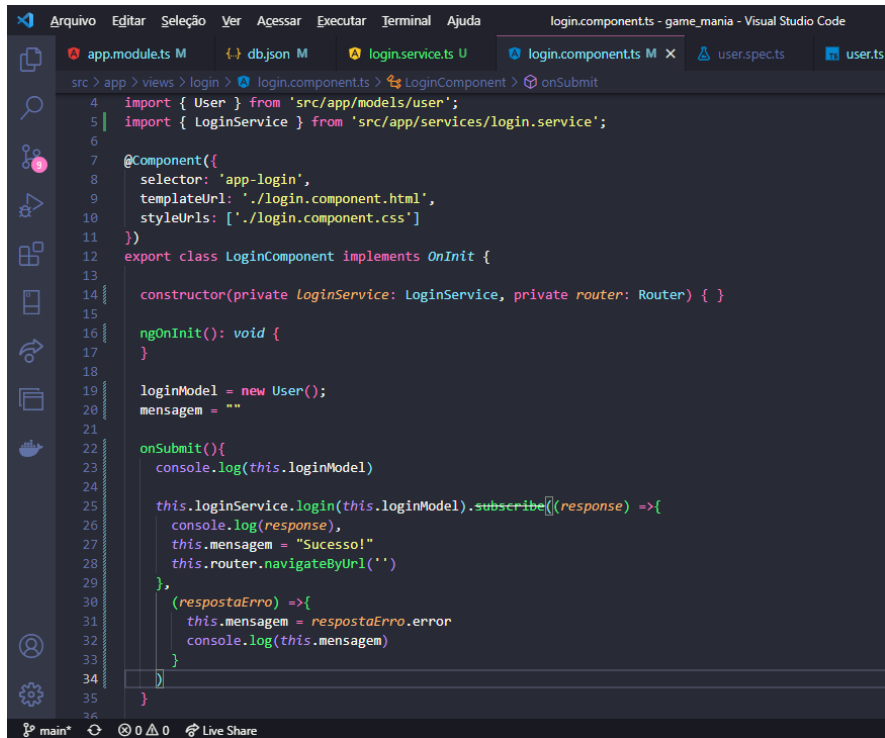
Ao preencher errado exibe mensagem:



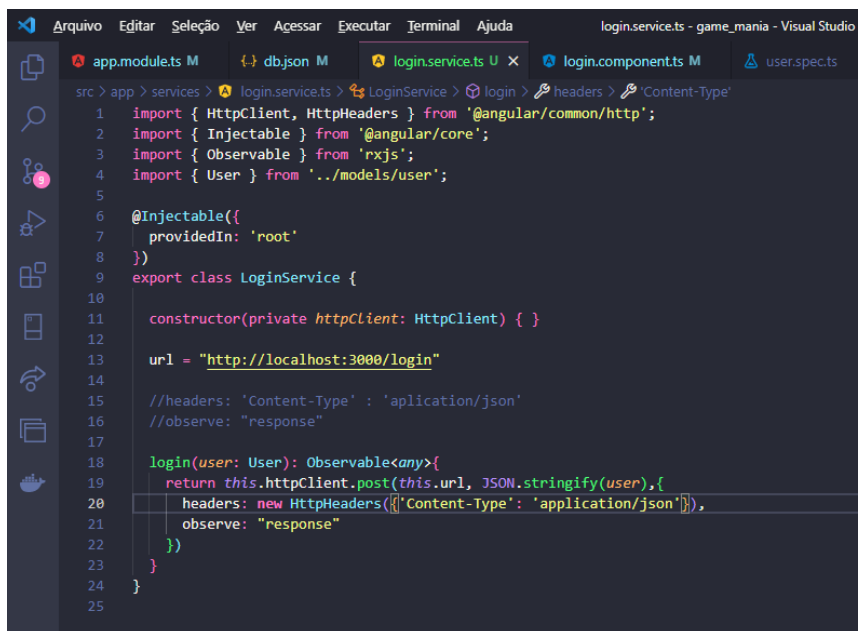




```
1 export class User {
2   constructor(
3     // public name: string = "",
4     public email: string = "",
5     public password?: string
6   ){
7   }
8 }
9
10
```



```
4 import { User } from 'src/app/models/user';
5 import { LoginService } from 'src/app/services/login.service';
6
7 @Component({
8   selector: 'app-login',
9   templateUrl: './login.component.html',
10  styleUrls: ['./login.component.css']
11 })
12 export class LoginComponent implements OnInit {
13
14   constructor(private loginService: LoginService, private router: Router) { }
15
16   ngOnInit(): void {
17   }
18
19   loginModel = new User();
20   mensagem = ""
21
22   onSubmit(){
23     console.log(this.loginModel)
24
25     this.loginService.login(this.loginModel).subscribe((response) =>{
26       console.log(response),
27       this.mensagem = "Sucesso!"
28       this.router.navigateByUrl('/')
29     }),
30     (respostaErro) =>{
31       this.mensagem = respostaErro.error
32       console.log(this.mensagem)
33     }
34   })
35 }
36
```



```
1 import { HttpClient, HttpHeaders } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { Observable } from 'rxjs';
4 import { User } from '../models/user';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class LoginService {
10
11   constructor(private httpClient: HttpClient) { }
12
13   url = "http://localhost:3000/login"
14
15   //headers: 'Content-Type': 'application/json'
16   //observe: "response"
17
18   login(user: User): Observable<any>{
19     return this.httpClient.post(this.url, JSON.stringify(user),{
20       headers: new HttpHeaders([{'Content-Type': 'application/json'}]),
21       observe: "response"
22     })
23   }
24 }
25
```

TESTE DE ATAQUE SQL injection

