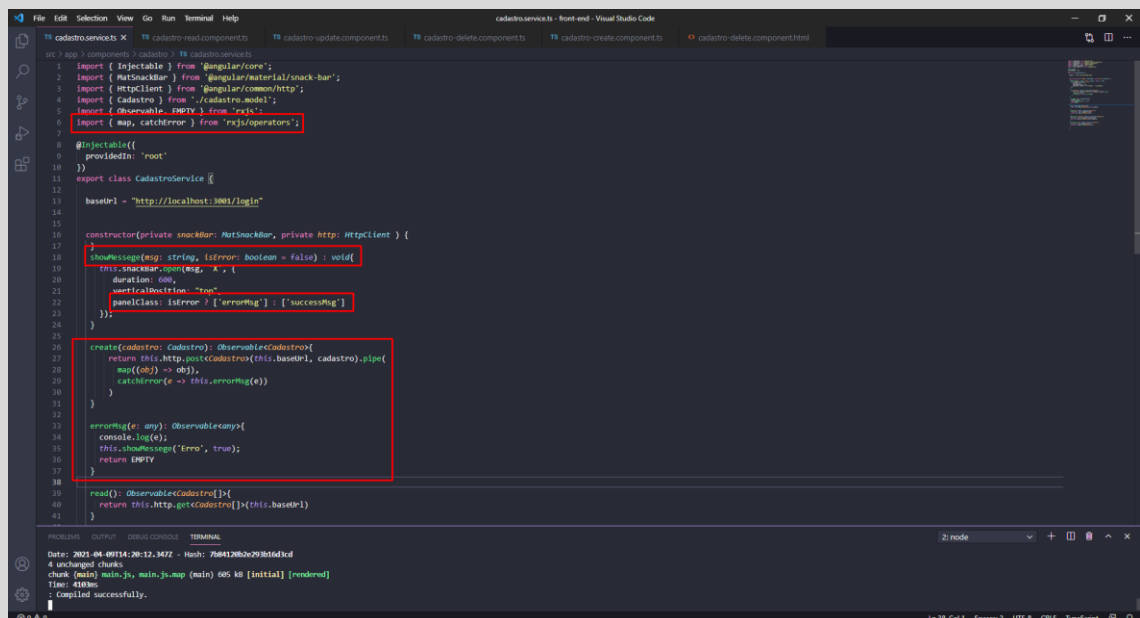


Configuração de mensagem de erro

1. No arquivo **cadastro.service.ts**, adicione as seguintes linhas de código:



```
1 import { Injectable } from '@angular/core';
2 import { MatSnackBar } from '@angular/material/snack-bar';
3 import { HttpClient } from '@angular/common/http';
4 import { Cadastro } from './cadastro.model';
5 import { Observable, throwError } from 'rxjs';
6 import { map, catchError } from 'rxjs/operators';
7
8 @Injectable({
9   providedIn: 'root'
10 })
11 export class CadastroService {
12   baseUrl = 'http://localhost:3001/login';
13
14   constructor(private snackBar: MatSnackBar, private http: HttpClient) {
15
16     showMessage(msg: string, isError: boolean = false) : void {
17       this.snackBar.open(msg, 'X', {
18         duration: 600,
19         verticalPosition: 'top',
20         panelClass: isError ? ['errorMsg'] : ['successMsg']
21       });
22     }
23
24     create(cadastro: Cadastro): Observable<Cadastro> {
25       return this.http.post<Cadastro>(this.baseUrl, cadastro).pipe(
26         map((obj) => obj),
27         catchError(e => this.handleError(e))
28       );
29     }
30
31     handleError(e: any): Observable<any> {
32       console.log(e);
33       this.showMessage('erro', true);
34       return throwError(e);
35     }
36
37     read(): Observable<Cadastro[]> {
38       return this.http.get<Cadastro[]>(this.baseUrl);
39     }
40
41   }
42 }
```

Na linha 18, o **showMessege** aparecerá como modal quando um conjunto de dados do seu projeto for inserido, atualizado ou excluído.

Além da mensagem de erro, existem mensagens de sucesso (por exemplo, “cadastro inserido”, “cadastro atualizado”, “cadastro deletado”) que podem ser configuradas dentro de cada arquivo **.ts** (cadastro-create.component.ts, cadastro-read.component.ts, cadastro-update.component.ts e cadastro-delete.component.ts).

No entanto, se o servidor não estiver conectado, um erro pode ocorrer, sendo necessário avisar o usuário, que não sabe se sua ação foi concluída ou não.

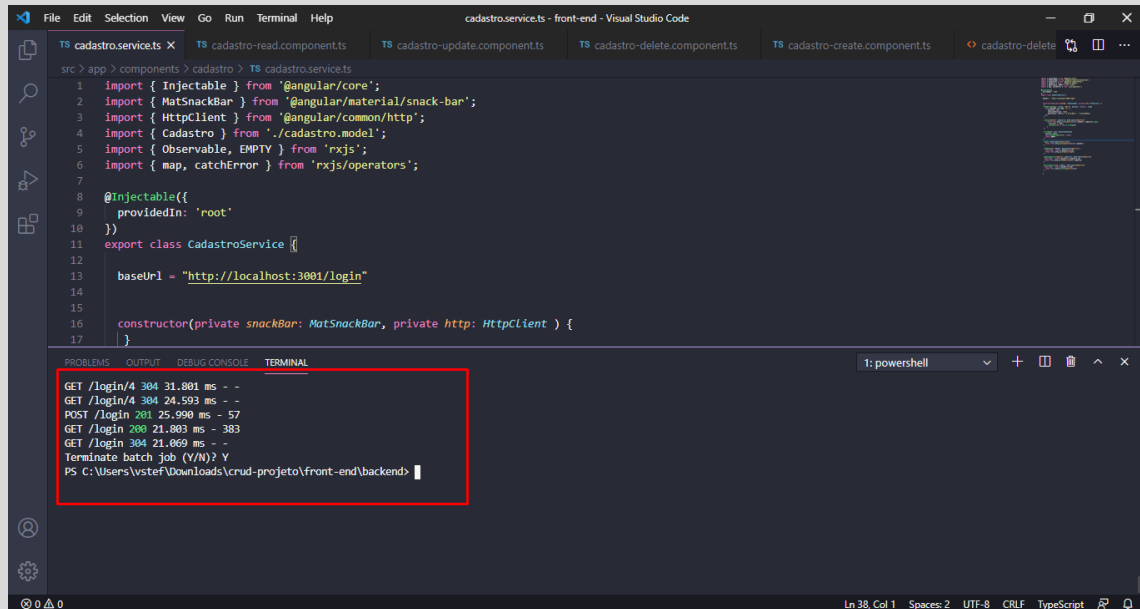
Importante

Lembre-se de que o usuário, provavelmente, não tem conhecimento técnico, logo, as informações devem ser claras e acessíveis. Em caso de erro, a maior parte dos usuários não saberá como acessar ou inspecionar o console, ou mesmo compreender o código, por isso, é imprescindível que nos atentemos às mensagens de erro.



Na linha 33, é aberto o conjunto de códigos que chamará o modal de erro. O **errorMsg** é uma variável cujo nome pode ser alterado e que é apresentada em caso de erro de Back-End. O **EMPTY** é um método que retorna um **Observable**. Na linha 29, estamos chamando o erro para que ele possa ser apresentado ao usuário.

2. Para testar, desligue seu servidor Back-End aberto no terminal (ctrl + c), mas deixe o servidor do Front-End funcionando.



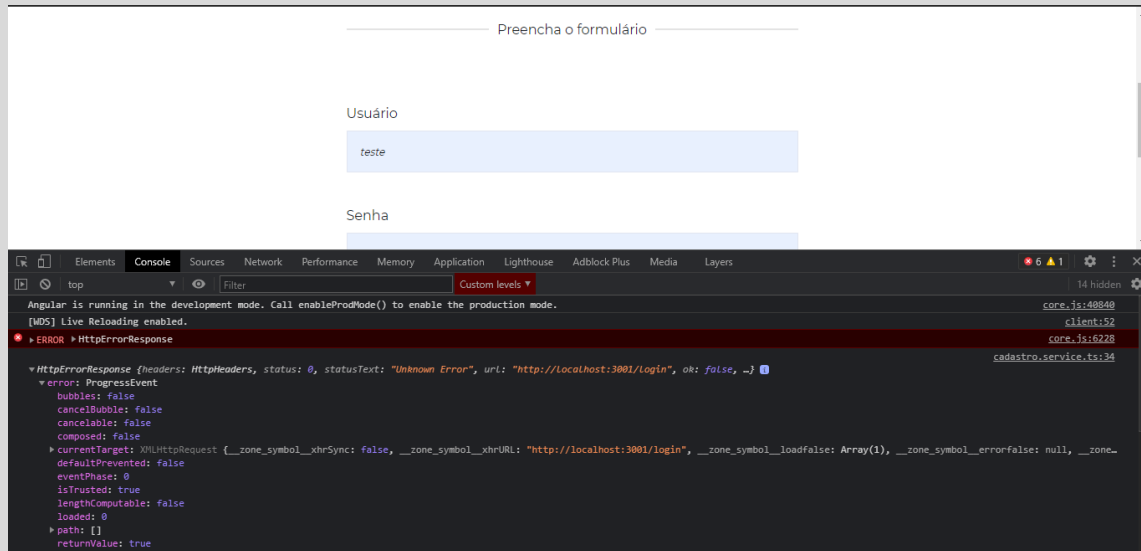
The screenshot shows the Visual Studio Code interface. The editor displays a TypeScript file named `cadastro.service.ts` with the following code:

```
1 import { Injectable } from '@angular/core';
2 import { MatSnackBar } from '@angular/material/snack-bar';
3 import { HttpClient } from '@angular/common/http';
4 import { Cadastro } from './cadastro.model';
5 import { Observable, EMPTY } from 'rxjs';
6 import { map, catchError } from 'rxjs/operators';
7
8 @Injectable({
9   providedIn: 'root'
10 })
11 export class CadastroService {
12
13   baseUrl = "http://localhost:3001/login"
14
15   constructor(private snackBar: MatSnackBar, private http: HttpClient) {
16
17   }
```

The terminal window at the bottom shows the output of a batch script, with a red box highlighting the last few lines:

```
GET /login/4 304 31.801 ms - -
GET /login/4 304 24.593 ms - -
POST /login 201 25.990 ms - 57
GET /login 200 21.803 ms - 383
GET /login 304 21.069 ms - -
Terminate batch job (Y/N)? Y
PS C:\Users\vstef\Downloads\crud-projeto\front-end\backend>
```

3. Em seu navegador, recarregue o browser (F5) e tente cadastrar um usuário. Observe o console:



O erro ocorre porque não há comunicação entre o Back-End e o Front-End.