

Relatório - Análise Estática

Este documento contém as etapas desenvolvidas durante a realização da atividade proposta de Análise Estática (Static Analysis).

1. Configurar o arquivo “**eslint.config.js**” para encontrar esses problemas e executar o comando “**npm run lint**”.

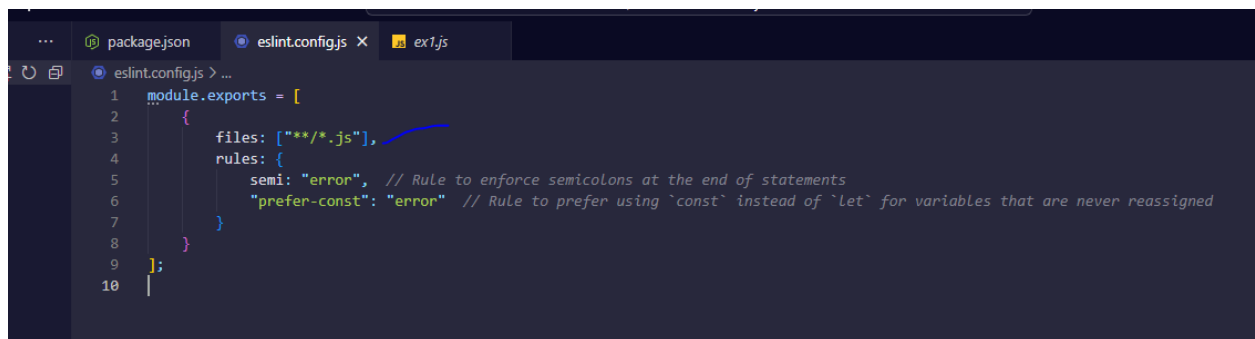
De primeiro momento, acessei a documentação do ESLint, disponível no site “<https://eslint.org/docs/latest/use/configure/configuration-files>”.

Criei o arquivo “**eslint.config.js**” e inseri o seguinte trecho:



```
1 module.exports = [
2   {
3     rules: {
4       semi: "error", // Rule to enforce semicolons at the end of statements
5       "prefer-const": "error" // Rule to prefer using `const` instead of `let` for variables that are never reassigned
6     }
7   }
8 ];
9
```

Também inseri a especificação das extensões dos arquivos que o **ESLint** irá validar:



```
1 module.exports = [
2   {
3     files: ["**/*.js"],
4     rules: {
5       semi: "error", // Rule to enforce semicolons at the end of statements
6       "prefer-const": "error" // Rule to prefer using `const` instead of `let` for variables that are never reassigned
7     }
8   }
9 ];
10
```

Após a inserção dessas configurações, executei pela primeira vez o comando “**npm run lint**” e obtive o seguinte resultado:

The screenshot shows the VS Code interface with the Explorer on the left showing the project structure: `node_modules`, `eslint.config.js`, `ex1.js`, `ex2.js`, `package-lock.json`, and `package.json`. The main editor displays the output of the command `npm run lint` in the terminal. The output shows 30 errors, all of which are 'Missing semicolon' (30 errors, 0 warnings). The errors are listed for two files: `ex1.js` and `ex2.js`. The errors are as follows:

File	Line	Column	Error	Suggestion	
C:\Users\03254244036\Documents\aula-publicacao-teste-apps-web\2-static-code-analysis\ex1.js	2:13	error	Missing semicolon	semi	
	3:9	error	Missing semicolon	semi	
	4:12	error	Missing semicolon	semi	
	7:39	error	Missing semicolon	semi	
	11:19	error	Missing semicolon	semi	
	15:11	error	Missing semicolon	semi	
	17:15	error	Missing semicolon	semi	
	21:43	error	Missing semicolon	semi	
	23:7	error	'obj' is never reassigned. Use 'const' instead	prefer-const	
	24:9	error	Missing semicolon	semi	
C:\Users\03254244036\Documents\aula-publicacao-teste-apps-web\2-static-code-analysis\ex2.js	1:30	error	Missing semicolon	semi	
	2:24	error	Missing semicolon	semi	
	3:26	error	Missing semicolon	semi	
	4:26	error	Missing semicolon	semi	
	7:34	error	Missing semicolon	semi	
	9:11	error	Missing semicolon	semi	
	12:11	error	Missing semicolon	semi	
	13:39	error	Missing semicolon	semi	
	16:37	error	Missing semicolon	semi	
	19:19	error	Missing semicolon	semi	
C:\Users\03254244036\Documents\aula-publicacao-teste-apps-web\2-static-code-analysis\ex2.js	23:15	error	Missing semicolon	semi	
	26:21	error	Missing semicolon	semi	
	29:23	error	Missing semicolon	semi	
	31:34	error	Missing semicolon	semi	
	33:5	error	'multiline' is never reassigned. Use 'const' instead	prefer-const	
	35:10	error	Missing semicolon	semi	
	37:23	error	Missing semicolon	semi	
	X 30 problems (30 errors, 0 warnings)				
	30 errors and 0 warnings potentially fixable with the '--fix' option.				

The terminal also shows the command `npm run lint` and the output of the command.

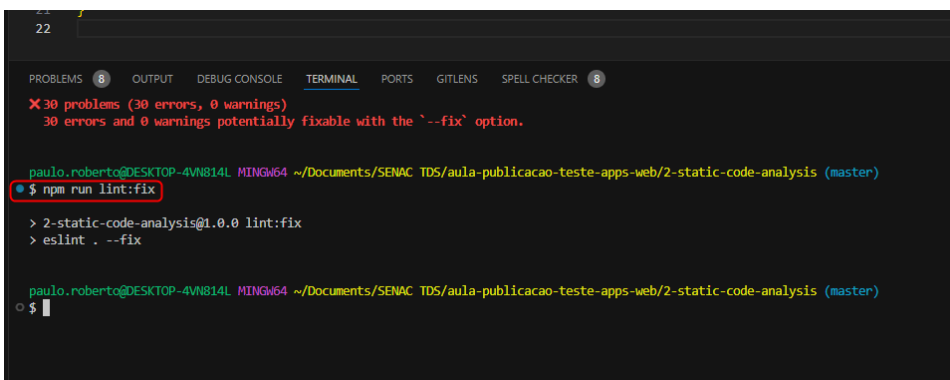
2. Criar script “ lint:fix ” no package.json para corrigir automaticamente

Para gerarmos esse script de correção, através da leitura na documentação, identifiquei que basta inserir a seguinte linha no script no arquivo “ **package.json** “:

The screenshot shows the VS Code interface with the Explorer on the left showing the project structure: `node_modules`, `eslint.config.js`, `ex1.js`, `ex2.js`, `package-lock.json`, and `package.json`. The main editor displays the `package.json` file. A red arrow points to the `lint:fix` script in the `scripts` section of the `package.json` file. The `package.json` file is as follows:

```
{
  "name": "2-static-code-analysis",
  "version": "1.0.0",
  "main": "ex1.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start1": "node ex1.js",
    "start2": "node ex1.js",
    "lint": "eslint .",
    "lint:fix": "eslint . --fix"
  },
  "author": "Paulo Roberto Xavier da Silva",
  "license": "ISC",
  "description": "",
  "devDependencies": {
    "@eslint/js": "^9.15.0",
    "eslint": "^9.15.0",
    "globals": "^15.12.0"
  }
}
```

Feito esse ajuste, abri o terminal no VSCode e apliquei o comando “ **npm run lint:fix** ”



```
21
22

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 8

X 30 problems (30 errors, 0 warnings)
  30 errors and 0 warnings potentially fixable with the '--fix' option.

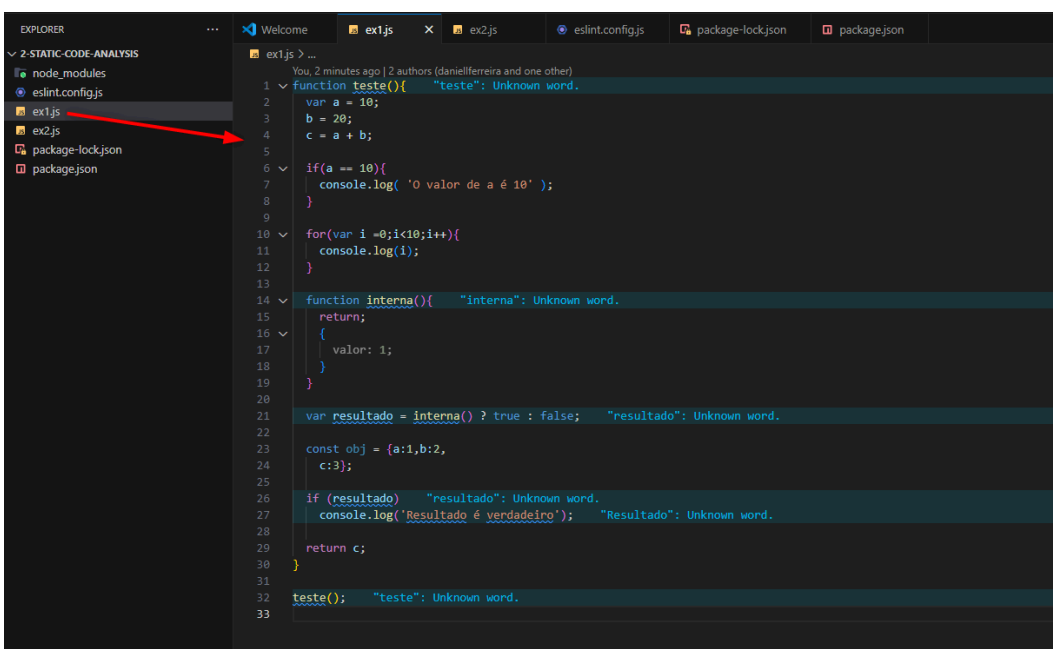
paulo.roberto@DESKTOP-4VN814L MINGW64 ~/Documents/SENAC TDS/aula-publicacao-teste-apps-web/2-static-code-analysis (master)
$ npm run lint:fix

> 2-static-code-analysis@1.0.0 lint:fix
> eslint . --fix

paulo.roberto@DESKTOP-4VN814L MINGW64 ~/Documents/SENAC TDS/aula-publicacao-teste-apps-web/2-static-code-analysis (master)
$
```

Após executar o comando, observei que os arquivos “ **ex1.js** ” e “ **ex2.js** ” foram modificados, conforme as regras que eu inseri no arquivo “ **eslint.config.js** ”.

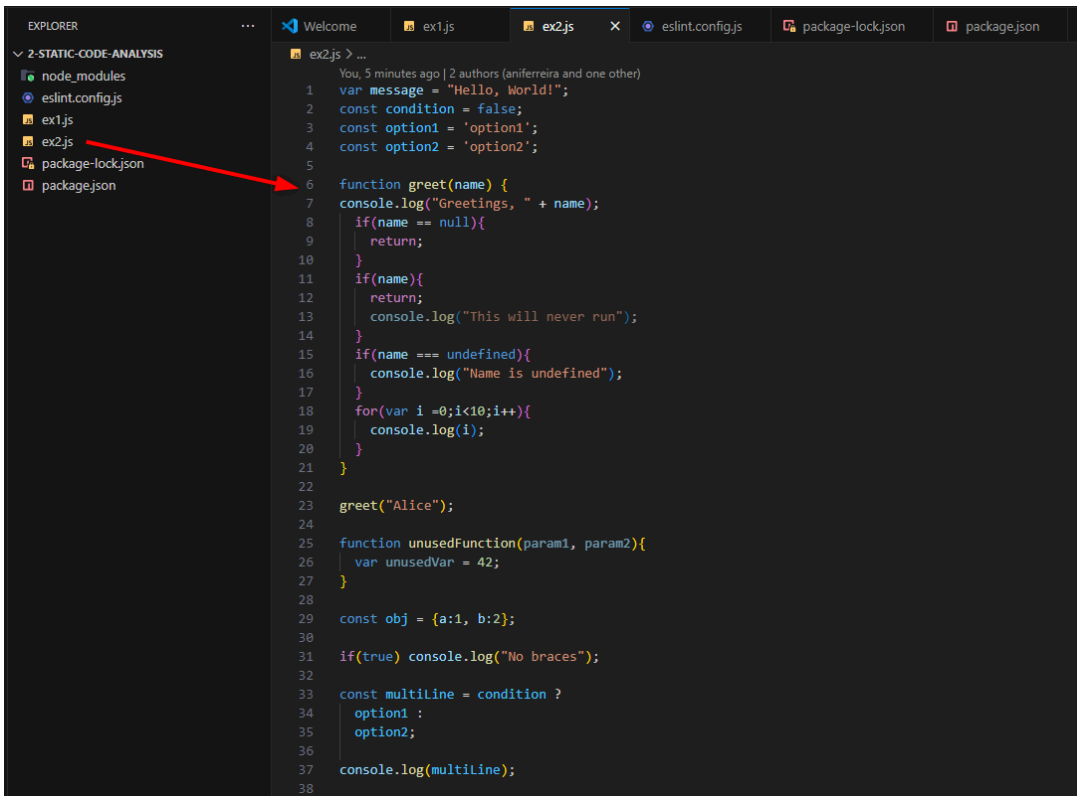
ex1.js



```
EXPLORER
2-STATIC-CODE-ANALYSIS
node_modules
eslint.config.js
ex1.js
ex2.js
package-lock.json
package.json

ex1.js > ...
You, 2 minutes ago | 2 authors (danielferreira and one other)
1 function teste(){ "teste": Unknown word.
2   var a = 10;
3   b = 20;
4   c = a + b;
5
6   if(a == 10){
7     console.log( '0 valor de a é 10' );
8   }
9
10  for(var i =0;i<10;i++){
11    console.log(1);
12  }
13
14  function interna(){ "interna": Unknown word.
15    return;
16    {
17      valor: 1;
18    }
19  }
20
21  var resultado = interna() ? true : false; "resultado": Unknown word.
22
23  const obj = {a:1,b:2,
24    c:3};
25
26  if (resultado) "resultado": Unknown word.
27    console.log('Resultado é verdadeiro'); "Resultado": Unknown word.
28
29  return c;
30
31
32 teste(); "teste": Unknown word.
33
```

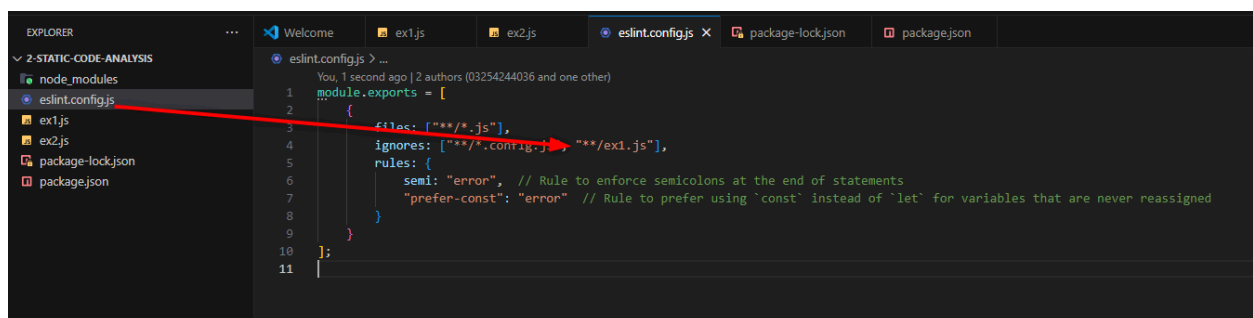
ex2.js



```
1  var message = "Hello, World!";
2  const condition = false;
3  const option1 = 'option1';
4  const option2 = 'option2';
5
6  function greet(name) {
7    console.log("Greetings, " + name);
8    if(name == null){
9      return;
10   }
11   if(name){
12     return;
13     console.log("This will never run");
14   }
15   if(name === undefined){
16     console.log("Name is undefined");
17   }
18   for(var i =0;i<10;i++){
19     console.log(i);
20   }
21 }
22
23 greet("Alice");
24
25 function unusedFunction(param1, param2){
26   var unusedVar = 42;
27 }
28
29 const obj = {a:1, b:2};
30
31 if(true) console.log("No braces");
32
33 const multiline = condition ?
34   option1 :
35   option2;
36
37 console.log(multiline);
38
```

3. Desabilitar o linter um arquivo inteiro de cada vez para ter os resultados por arquivo.

Para ignorarmos determinados arquivos durante a execução do linter, basta inserirmos a seguinte linha no arquivo “**eslint.config.js**”, juntamente com o arquivo que desejamos ser ignorado:



```
1  module.exports = [
2    {
3      files: ["**/*.js"],
4      ignores: ["**/*.config.js", "**/ex1.js"],
5      rules: {
6        semi: "error", // Rule to enforce semicolons at the end of statements
7        "prefer-const": "error" // Rule to prefer using `const` instead of `let` for variables that are never reassigned
8      }
9    }
10 ];
11
```

Para fazer o teste, visto que com o enunciado anterior já foram corrigidos os erros, acessei o arquivo “**ex2.js**” e removi alguns pontos e vírgulas para que sejam exibidos erros.

Abri o terminal e novamente inseri o comando “**npm run lint**”.

```
1 var message = "Hello, World!"
2 const condition = false
3 const option1 = 'option1'
4 const option2 = 'option2'

5
6 function greet(name) {
7   console.log("Greetings, " + name);
8   if(name == null){
9     return;
10  }
11  if(name){
12    return;
13    console.log("This will never run");
14  }
15  if(name === undefined){
16    console.log("Name is undefined");
17  }
18  for(var i =0;i<10;i++){
19    console.log(i);
20  }
21 }
```

```
> 2-static-code-analysis@1.0.0 lint
> eslint .

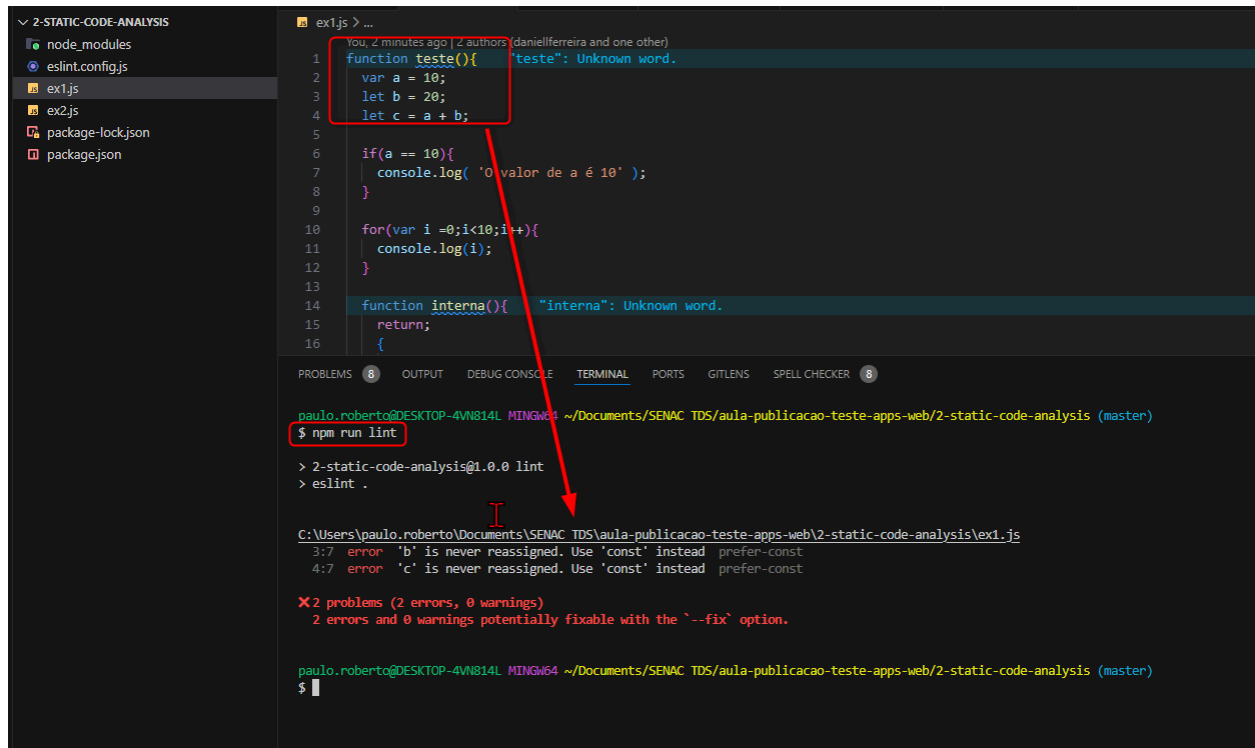
C:\Users\paulo.roberto\Documents\SENAC TDS\aula-publicacao-teste-apps-web\2-static-code-analysis\ex2.js
1:30 error  Missing semicolon  semi
2:24 error  Missing semicolon  semi
3:26 error  Missing semicolon  semi
4:26 error  Missing semicolon  semi

✖ 4 problems (4 errors, 0 warnings)
4 errors and 0 warnings potentially fixable with the `--fix` option.

paulo.roberto@DESKTOP-4VN814L MINGW64 ~/Documents/SENAC TDS/aula-publicacao-teste-apps-web/2-static-code-analysis (master)
$
```

Agora, refiz o mesmo procedimento, porém para que seja ignorado o arquivo “ ex2.js “:

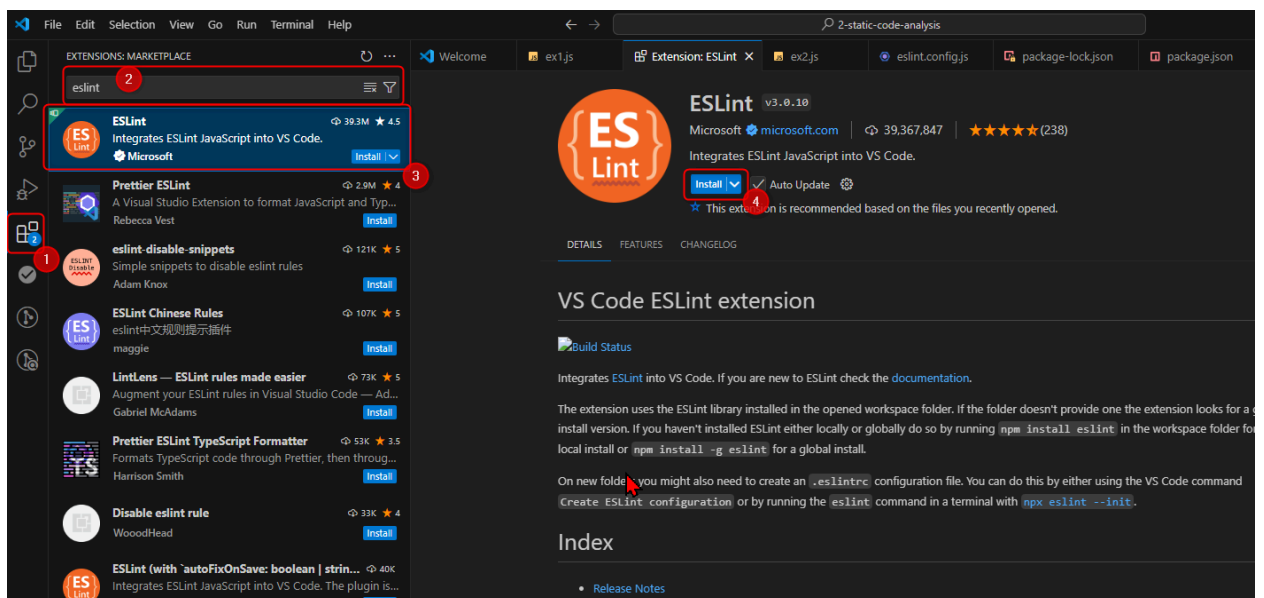
```
1 module.exports = [
2   {
3     files: ["**/*.js"],
4     ignores: ["**/*.config.js", "**/ex2.js"],
5     rules: {
6       semi: "error", // Rule to enforce semicolons at the end of statements
7       "prefer-const": "error" // Rule to prefer using `const` instead of `let` for variables that are never reassigned
8     }
9   }
10 ];
11
```



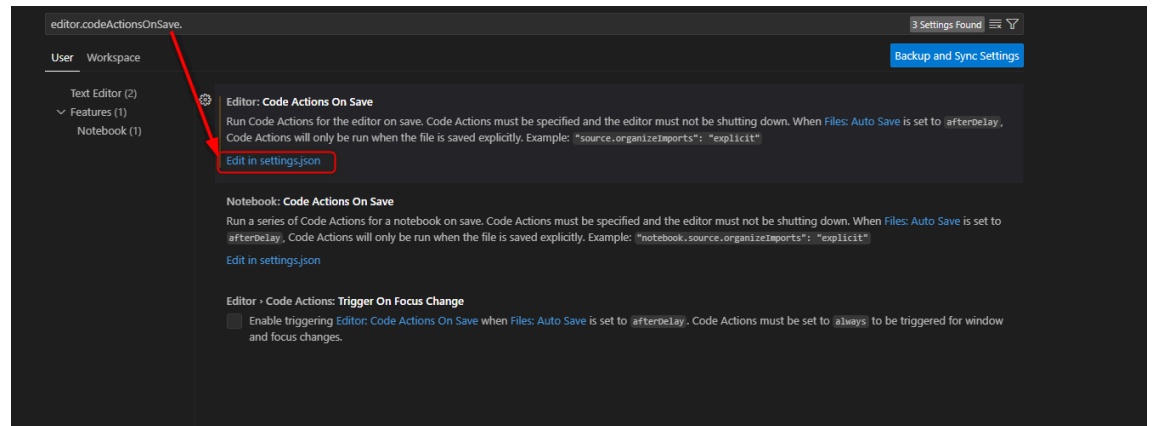
4. Configurar o editor para corrigir automaticamente ao salvar

Para fazer essas configurações, apliquei a seguinte sequência:

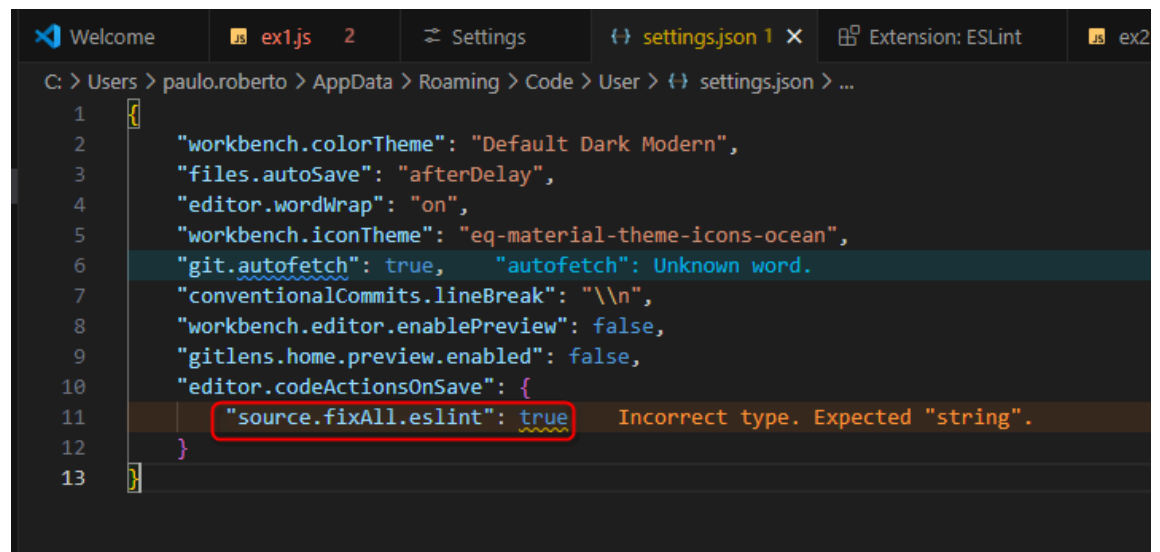
- Abri o ícone de “**Extensões**” do VSCode e instalei a extensão oficial da Microsoft chamada “**ESLint**”;



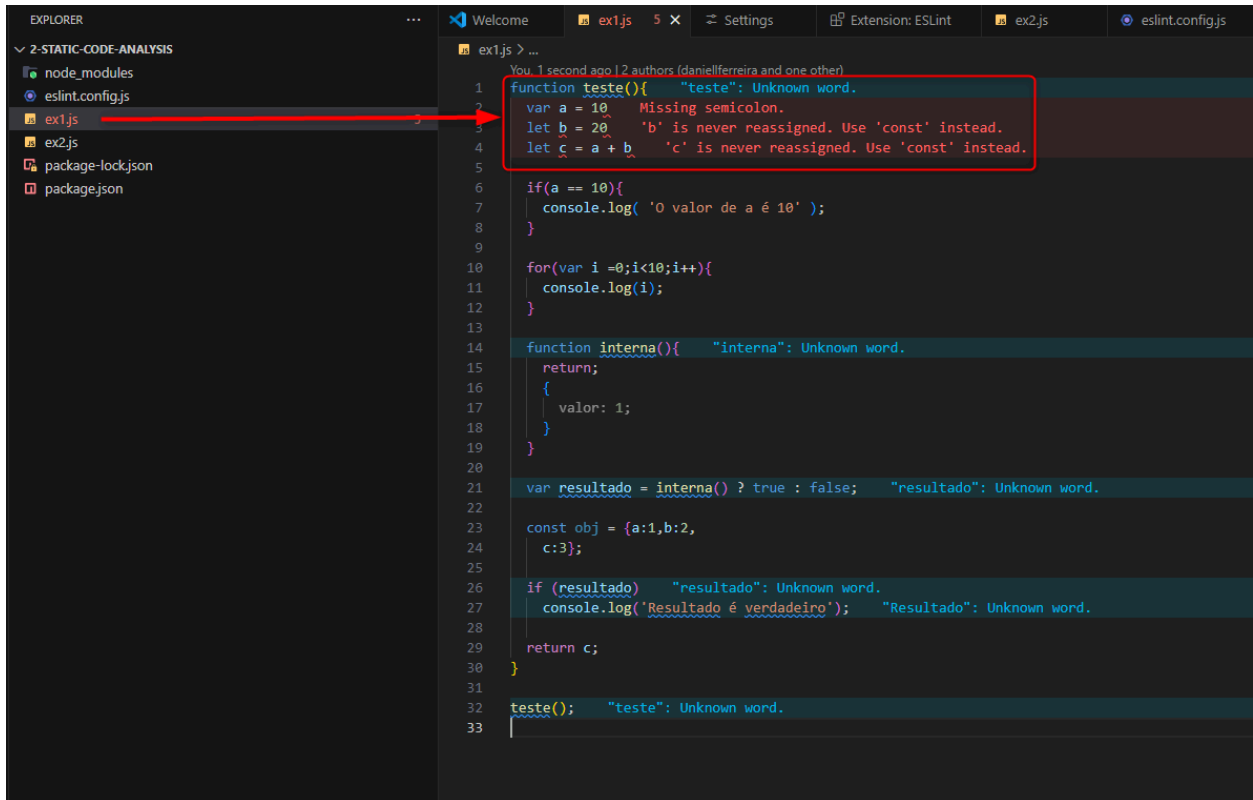
- Posteriormente, acessei as configurações do VSCode através do atalho “**Ctrl+,”**
- Pesquisei por “**editor.codeActionsOnSave**” e selecionei a opção “**Edit in settings.json**”



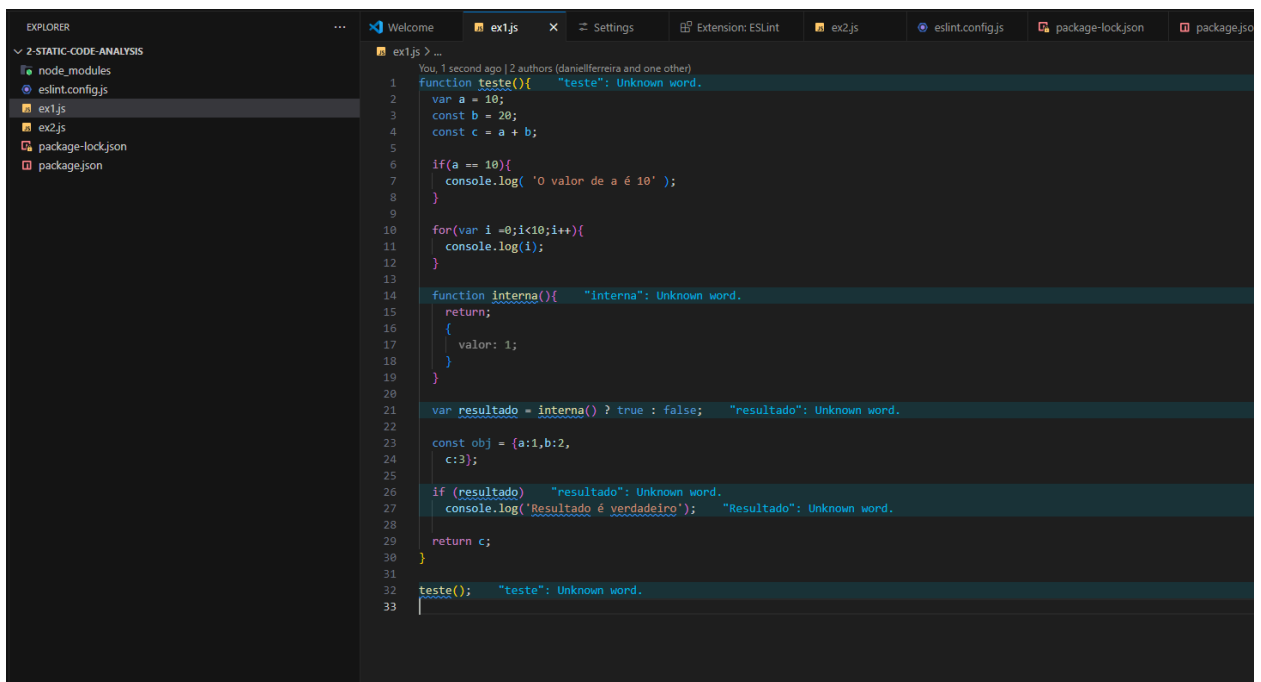
- Por fim, inseri a seguinte linha: “**source.fixAll.eslint**” : true



Para fazer o teste e validar, acessei o arquivo “**ex1.js**”, que estava constando alguns erros, conforme a imagem:

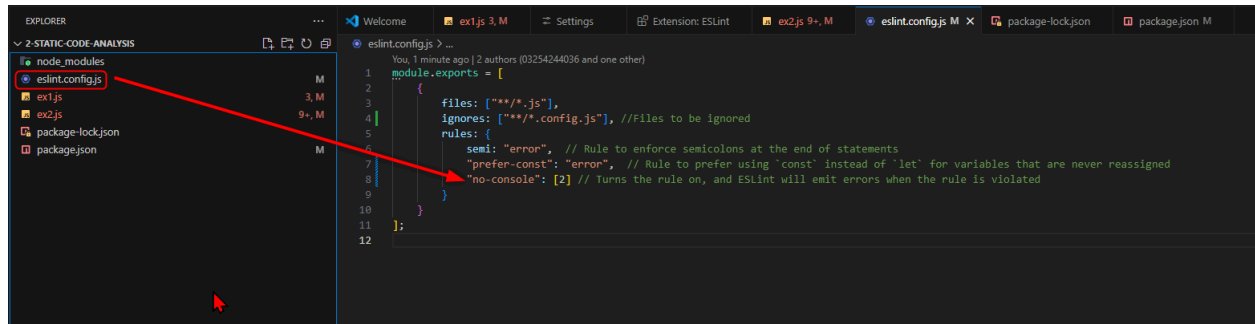


Pressionei “**Ctrl + s**” e automaticamente foi corrigido o código:

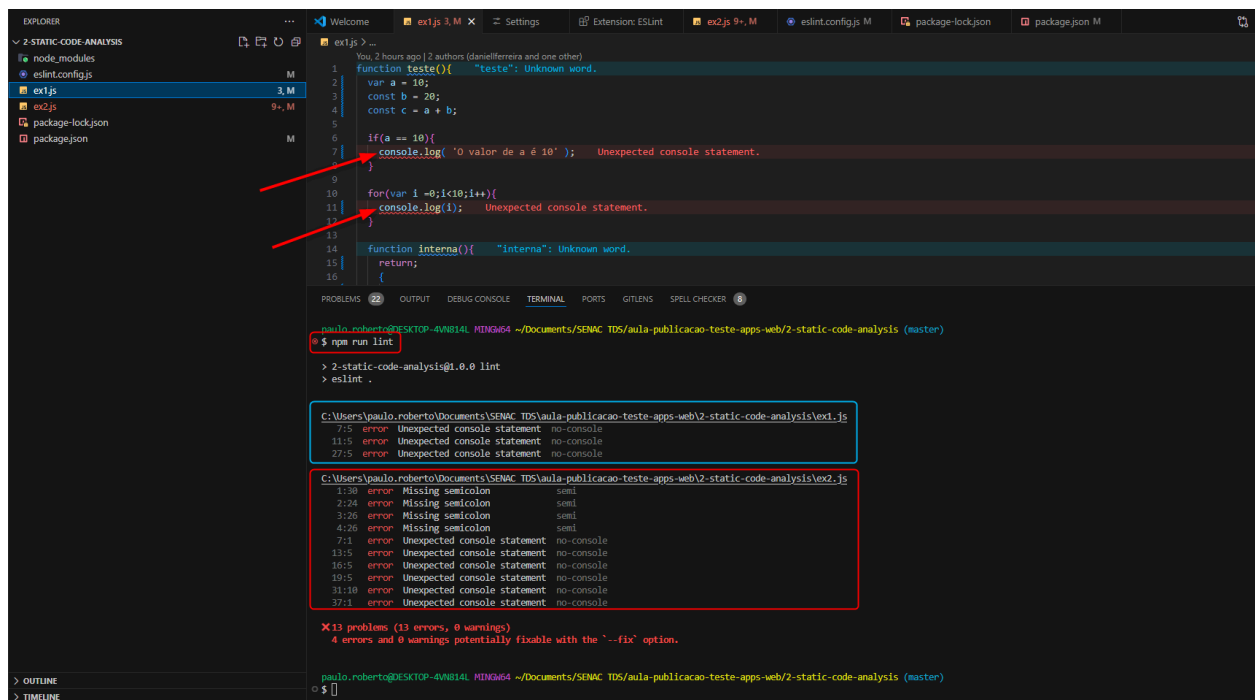


5. Desabilitar por linha os comandos “ **console.log** ”

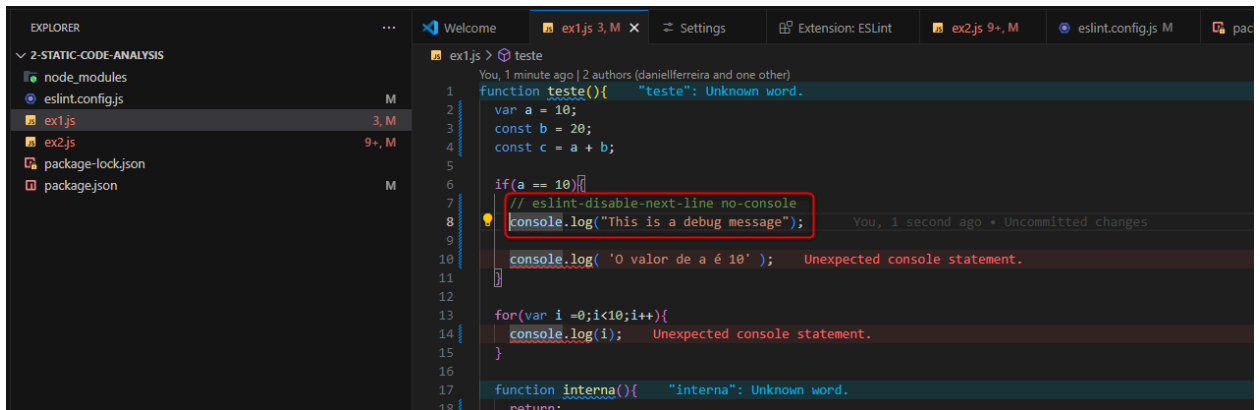
Para poder desabilitar os comandos de console, de primeiro momento, criei uma regra no arquivo “ **eslint.config.js** ” que bloqueie e emita erros ao utilizar o comando “ **console.log** ”



Após aplicada essa regra, executei novamente o comando “ **npm run lint** ” e obtive os seguintes erros, incluindo os erros de “ **console** ”:



Para fazer com que o ESLint ignore determinada linha contendo console, inseri o seguinte comentário antes da linha contendo o console: “ **// eslint-disable-next-line no-console** ”



Isso fez com que a mensagem de erro não fosse exibida, mesmo havendo uma regra aplicada dentro do arquivo “**eslint.config.js**”.

Também é possível configurar essa regra por bloco, conforme a imagem:

