

Trabalho de Qualidade e Teste

Descrição do Sistema

O Tabela-verdade é um gerador de tabela-verdade de expressões lógicas *open source* criado com React e Gatsby para *web*. A aplicação foi criada para a disciplina de Fundamentos Matemáticos para Computação, no 1º período do curso de Sistemas de Informação, da Universidade Federal Fluminense (UFF). Atualmente, esse objetivo se estendeu e agora está disponível para qualquer pessoa utilizá-lo. A aplicação é formada por quatro módulos principais: source, lexer, parser e generator. Esses módulos operam em sequência para transformar uma expressão lógica arbitrária de entrada em uma tabela-verdade na saída. Pretendemos testar cada um dos módulos e suas dependências unitariamente, testar a integração entre eles e por fim testar a aplicação em seu caso de uso típico.

Medidas dos atributos de qualidade

Escala de 0 a 10.

Qualidade do produto

- Adequação Funcional
 - Atribuição: 5
 - Justificativa: Como esse projeto não foi movido por fins comerciais, não tivemos um cliente externalizando suas necessidades. Entretanto, alguns requisitos foram levantados que o produto final deveria, no mínimo, ter,
- Eficiência de desempenho
 - Atribuição: 8
 - Justificativa: dado que a aplicação é executada inteiramente no dispositivo do usuário final, não temos o controle sobre a configuração do ambiente de execução, portanto gostaríamos que a aplicação tivesse um desempenho satisfatório em ambientes com recursos computacionais limitados.
- Compatibilidade
 - Atribuição: 3
 - Justificativa: Considerando que o aplicativo é um *stand alone* que tem um objetivo específico, não buscamos compatibilidade com outros sistemas.
- Usabilidade

- Atribuição:9
- Justificativa: Como a aplicação busca fornecer de forma rápida um resultado de uma operação matemática com um certo grau de complexidade, a Usabilidade se vê como um dos pilares do projeto; Usando diversos símbolos que encontramos na literatura oferecendo para o usuário final adequabilidade aos padrões matemáticos, mas também uma facilidade de uso e de aprendizado do sistema.
- Confiabilidade
 - Atribuição: 10
 - Justificativa: Por se tratar de um aplicativo acadêmico, busca-se garantir o menor número possível de estados de falha, e que a aplicação fique disponível o maior tempo possível.
- Segurança
 - Atribuição: 3
 - Justificativa: a aplicação não é projetada para realizar nenhuma troca de informação com ambientes externos ou manter quaisquer tipos de estado sobre sua execução e informações inseridas durante esse processo; portanto a segurança não deve ser um requisito estritamente priorizado.
- Manutenibilidade
 - Atribuição: 10
 - Justificativa: sendo *open source*, é fundamental que a aplicação tenha o código fonte altamente analisável e modificável, facilitando o entendimento por parte de terceiros interessados em estudá-lo e/ou contribuir diretamente para o seu desenvolvimento.
- Portabilidade
 - Atribuição: 8
 - Justificativa: Considerando o aplicativo como uma aplicação web, busca-se que o mesmo seja compatível com o maior número de OS e Navegadores possíveis, buscando alienar o menor número possível de usuários finais.

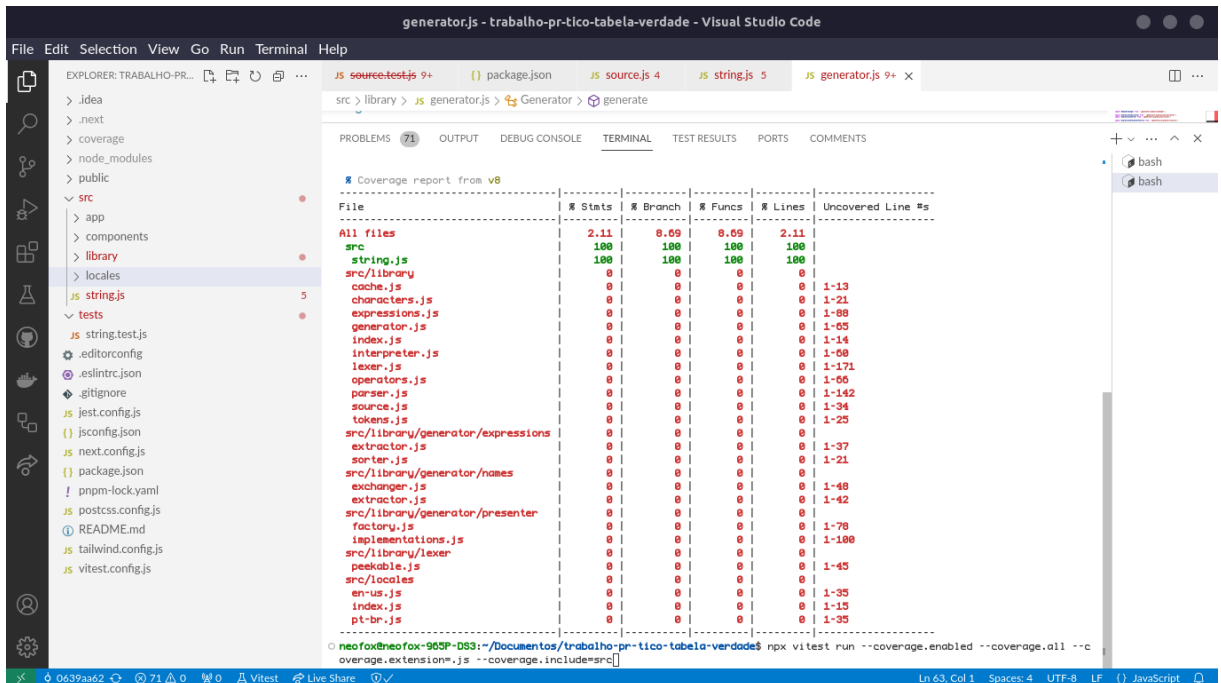
Qualidade em uso

- Eficácia
 - Atribuição: 10
 - Justificativa: Por se tratar de um domínio matemático, a aplicação deve sempre ser a mais precisa possível, erros de output não são tolerados.
- Eficiência
 - Atribuição: 6

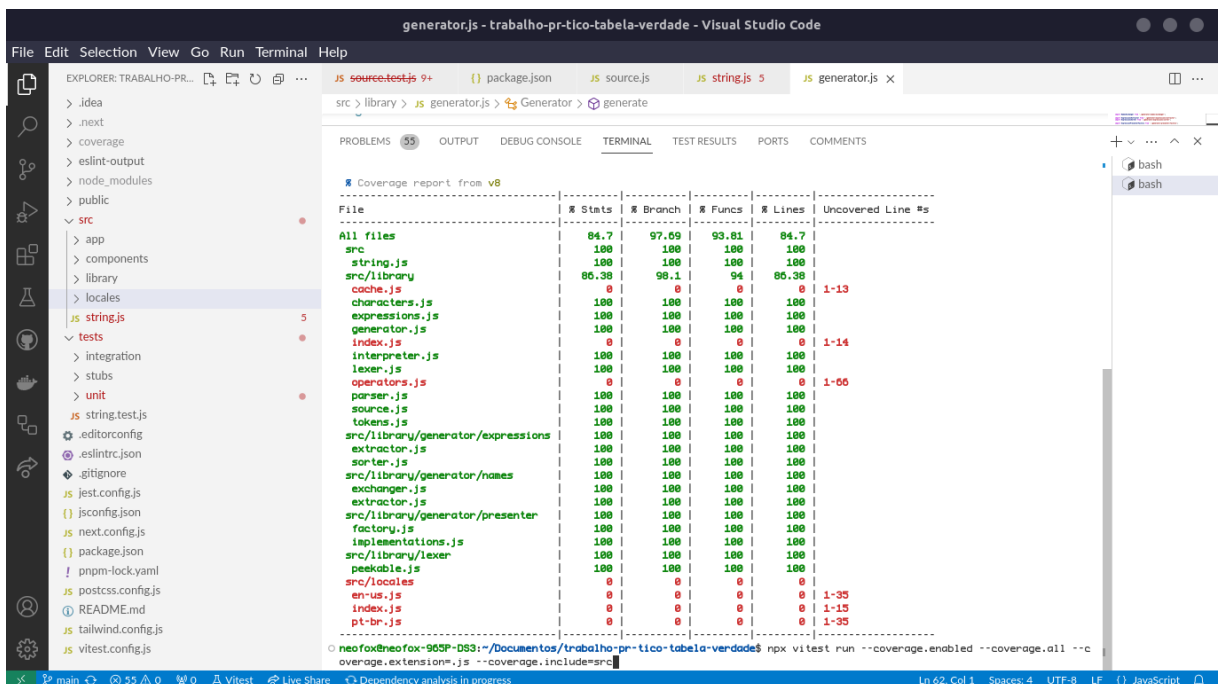
- Justificativa: Tratando-se de um aplicação web, entende-se que devemos nos preocupar com a Eficiência, mas ao mesmo tempo oferecer algum grau de leniência no executar da aplicação.
- Satisfação
 - Atribuição: 9
 - Justificativa: Como o sistema busca facilitar a vida do usuário final nesse domínio, entende-se que a satisfação do mesmo é inegociável.
- Safety
 - Atribuição: 1
 - Justificativa: A aplicação não lida com dados sensíveis, credenciais ou ameaças diretas para o usuário final. Por tanto, não existe nenhum risco intrínseco a sua utilização.
- Usabilidade
 - Atribuição:9
 - Justificativa: Como a aplicação busca fornecer de forma rápida um resultado de uma operação matemática com um certo grau de complexidade, a Usabilidade se vê como um dos pilares do projeto; Usando diversos símbolos que encontramos na literatura oferecendo para o usuário final adequabilidade aos padrões matemáticos, mas também uma facilidade de uso e de aprendizado do sistema.

Análise com o Vitest Coverage

O Vitest Coverage é uma opção para a ferramenta Vitest que gera um relatório da cobertura do código fonte por testes unitários. Utilizamos a ferramenta para tomarmos conhecimentos das partes do código fonte não testadas e então projetarmos testes unitários para as mesmas.



Antes (commit 0639aa62e88d78ffc0ec91e6b8413c7d3c50f339)



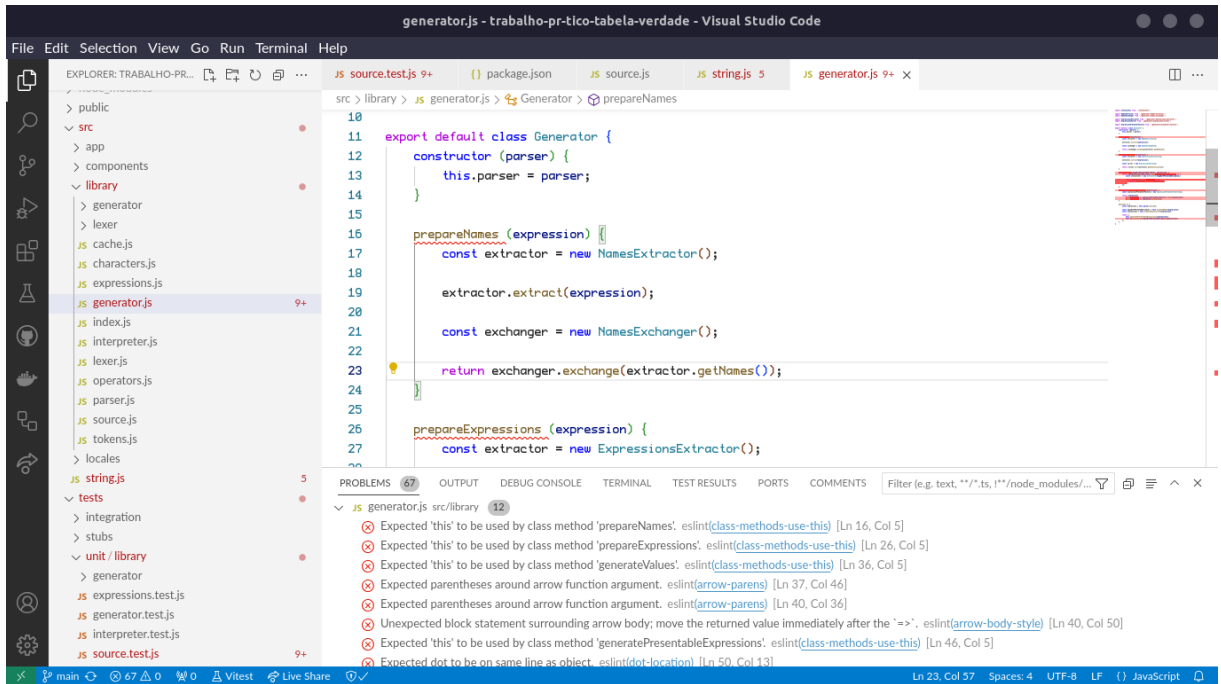
Depois (commit e25dc524d9c665f822c1b247b51429f01d2acf3d)

Análise com o ESLint

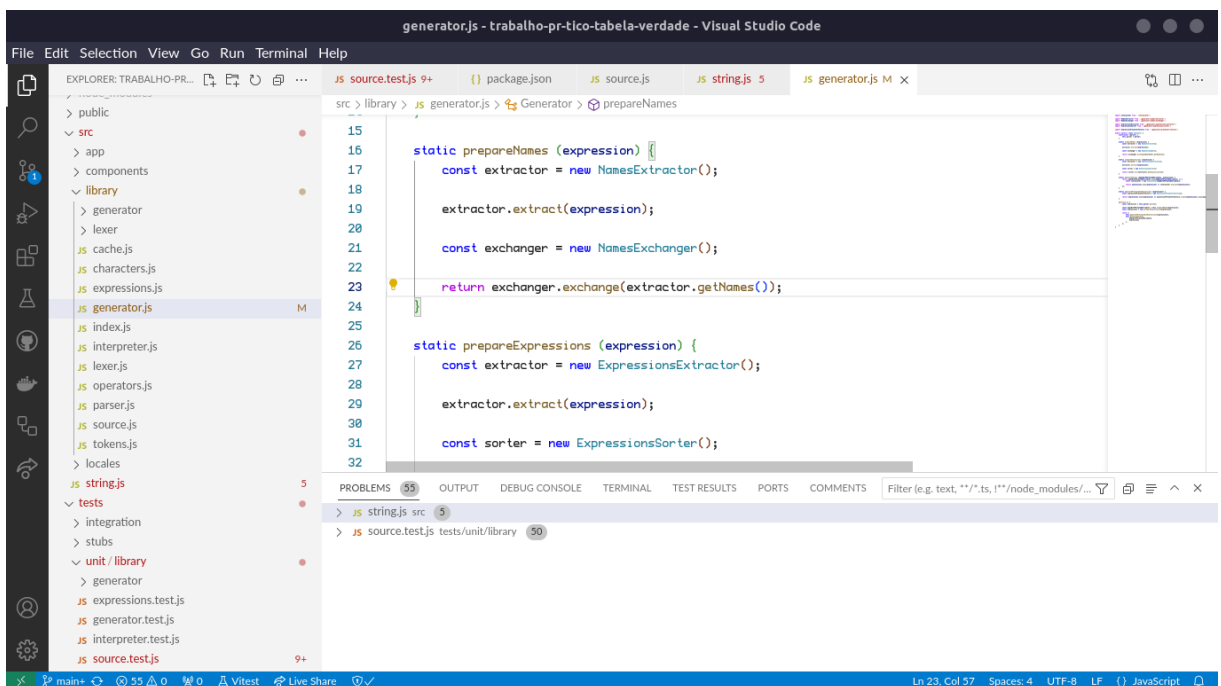
ESLint é uma ferramenta de análise de código fonte que reporta inconformidades na escrita do código fonte em relação às regras de escrita previamente estabelecidas para o projeto. Visando obter um estilo de escrita padronizado ao longo do código fonte de toda a

aplicação, refatoramos trechos de código conforme as inconformidades reportadas pela ferramenta.

Generator — Brendo Costa



Antes



Depois

Lexer — Mathews Reis

```
src > library > JS lexer.js > Lexer > next

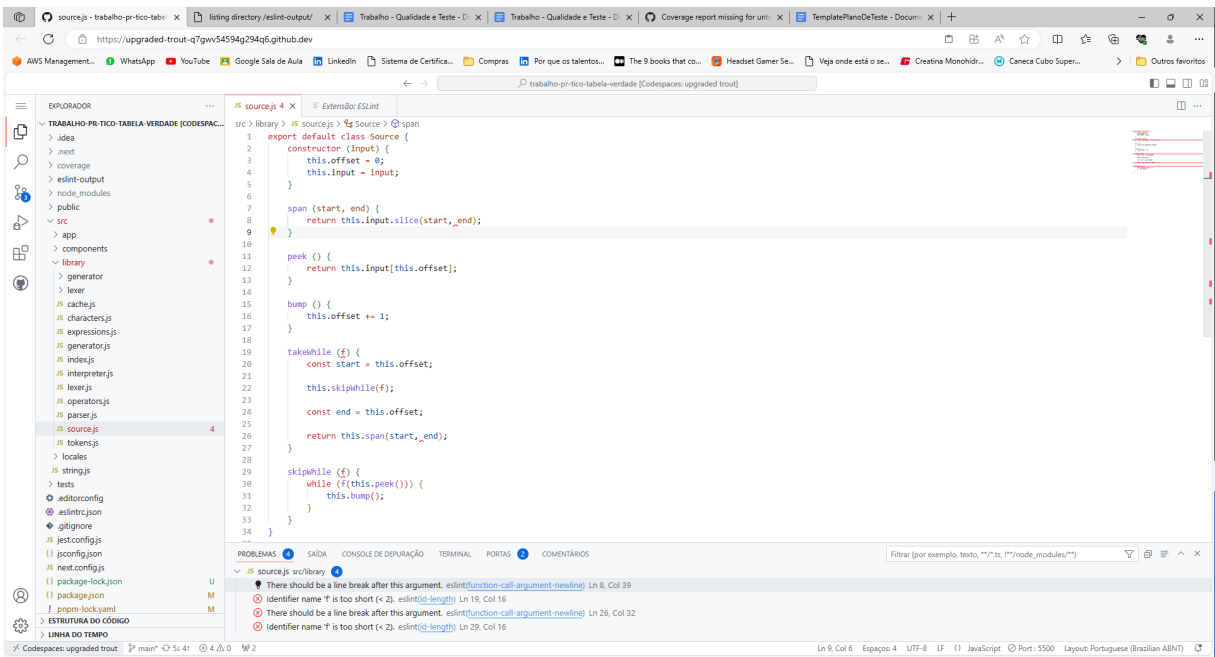
38 |     End,
39 | } from './tokens';
40 |
41 | export function isWhiteSpace (ch) {
42 |     return ch === '\n' // New line
43 |         || ch === '\r' // Return
44 |         || ch === '\t' // Tab
45 |         || ch === '\v' // Vertical tab
46 |         || ch === '\b' // Backspace
47 |         || ch === '\f' // Form feed
48 |         || ch === '\x20' // Space
49 |         || ch === '\xA0'; // No-break space
50 | }
51 |
52 | export function isNumber (ch) {
53 |     return (typeof ch === 'string' && ch.length === 1) && (ch >= '0' && ch <= '9');
54 | }
55 |
56 | export function isAlphabetic (ch) {
57 |     return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z');
58 | }
59 |
60 | export default class Lexer {
61 |     constructor (source) {
62 |         this.source = source;
63 |     }
64 |
65 |     next () {
66 |         this.source.skipWhile(isWhiteSpace);
67 |     }
68 | }
69 |
70 | export { Lexer };
71 |
72 | // Test
73 | const lexer = new Lexer(' ');
74 | lexer.next();
75 | console.log(lexer.source.current);
76 |
77 | ~~~~~
78 | case CARET:
79 | case ASTERISK:
80 | case AMPERSAND:
81 | case LOGICAL AND:
82 |     this.source.bump();
83 |
84 |     return new And();
85 |
86 | case PLUS:
87 | case VERTICAL LINE:
88 | case LOGICAL OR:
89 |     this.source.bump();
90 |
91 |     return new Or();
92 |
93 | case RIGHTWARDS ARROW:
94 | case RIGHTWARDS DOUBLE ARROW:
95 |     this.source.bump();
96 |
97 |     return new Conditional();
98 |
99 | case LEFT RIGHT ARROW:
100 | case LEFT RIGHT DOUBLE ARROW:
101 |     this.source.bump();
102 |
103 |     return new Biconditional();
104 |
105 | case OPENING PARENTHESIS:
106 |     this.source.bump();
107 |
108 | ~~~~~
109 | }
```

Antes

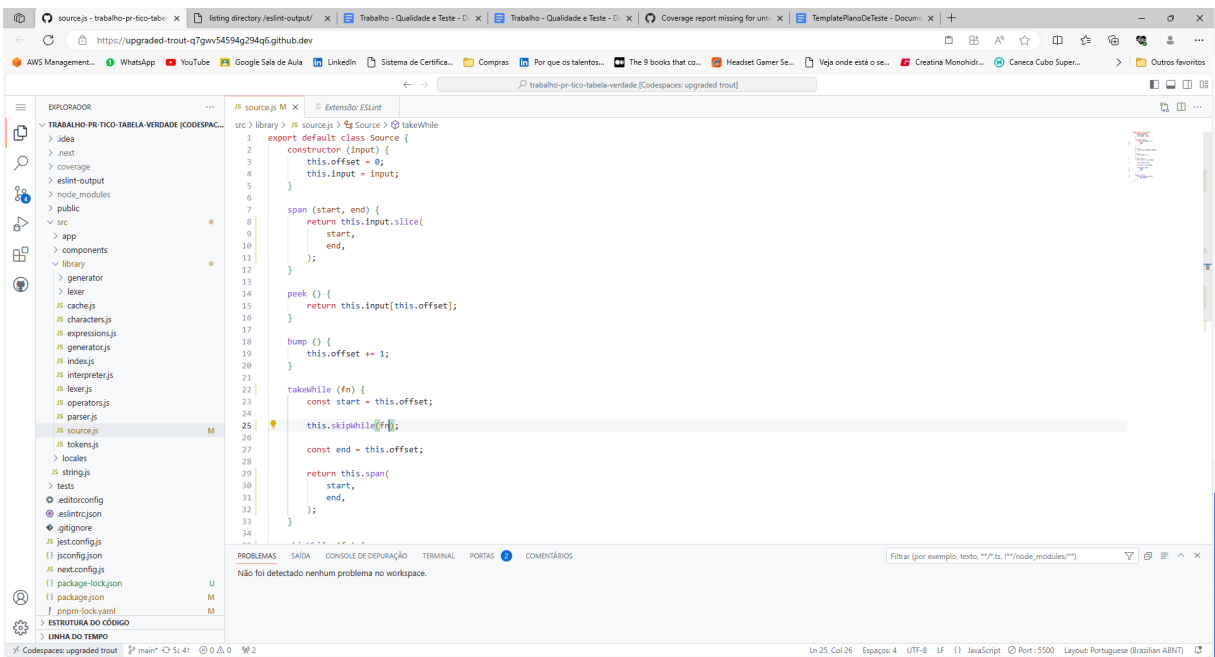
```
[Preview] README.md JS lexer.test.js 9+, U JS lexer.js 3, M X .eslintrc.json M JS source.js 4
src > library > JS lexer.js > ...
40
41 export function isWhiteSpace (ch) {
42     // New line
43     return ch === '\n' ||
44         // Return
45         ch === '\r' ||
46         // Return
47         ch === '\t' ||
48         // Tab
49         ch === '\v' ||
50         // Vertical tab
51         ch === '\b' ||
52         // Backspace
53         ch === '\f' ||
54         // Form feed
55         ch === '\x20' ||
56         // Space
57         ch === '\xA0';
58     // No-break space
59 }
60
61 next() {
62     this.source.skipWhile(isWhiteSpace);
63
64     const ch = this.source.peak();
65
66     switch (ch) {
67         case TILDE:
68         case EXCLAMATION:
69         case LOGICAL_NOT:
70             this.source.bump();
71
72             return new Not();
73
74         case CARET:
75         case ASTERISK:
76         case AMPERSAND:
77         case LOGICAL_AND:
78             this.source.bump();
79
80             return new And();
81
82         case PLUS:
83         case VERTICAL_LINE:
84         case LOGICAL_OR:
85             this.source.bump();
86
87             return new Or();
88
89         case RIGHTWARDS_ARROW:
90         case RIGHTWARDS_DOUBLE_ARROW:
91             this.source.bump();
92     }
93 }
94
95
96
97
98
99
100
101
102
103
104
105
0 1 Ln 125, Col 21 Espaços: 4 UTF-8 LF ( ) JavaScript Layout: Portuguese
```

Depois

Source — Isaac Luiz

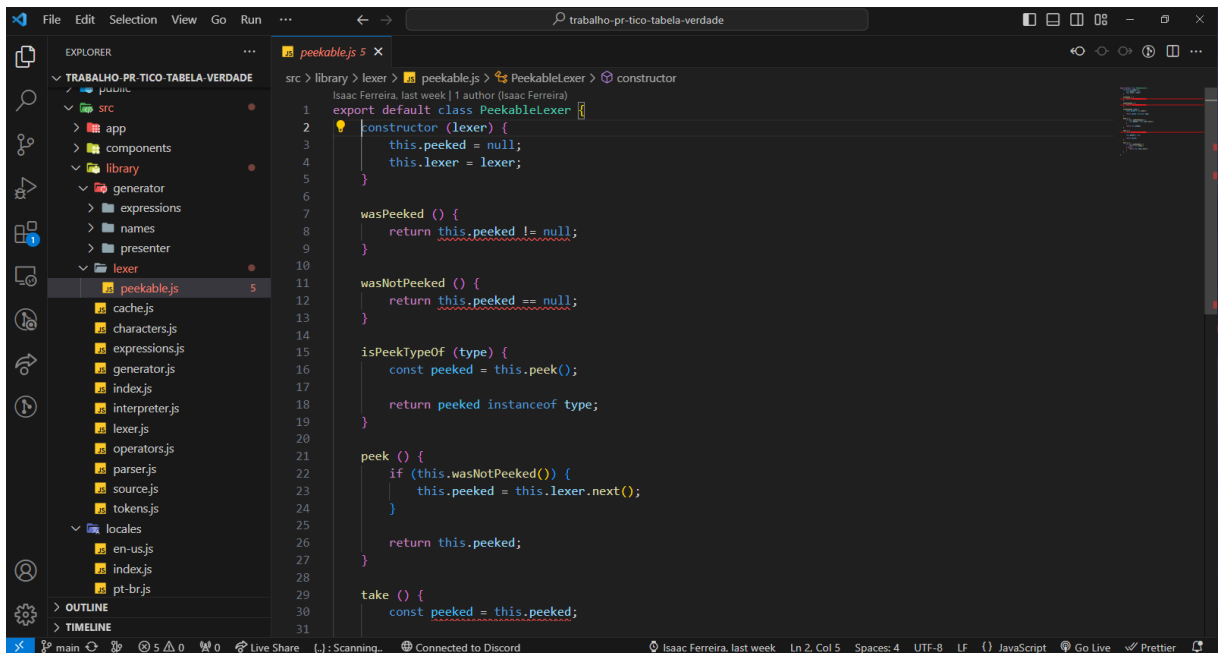


Antes



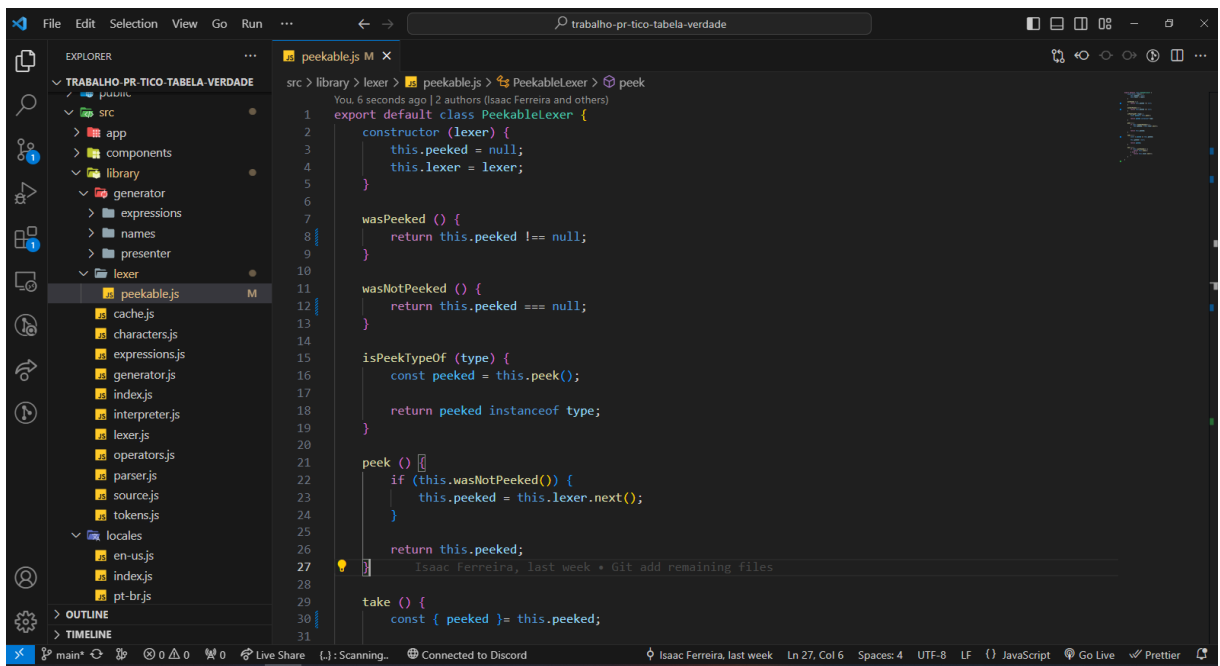
Depois

Peekable — Daniel Lima



```
1 export default class PeekableLex {
2   constructor (lexer) {
3     this.peeked = null;
4     this.lexer = lexer;
5   }
6
7   wasPeeked () {
8     return this.peeked != null;
9   }
10
11   wasNotPeeked () {
12     return this.peeked == null;
13   }
14
15   isPeekTypeOf (type) {
16     const peeked = this.peek();
17     return peeked instanceof type;
18   }
19
20   peek () {
21     if (this.wasNotPeeked()) {
22       this.peeked = this.lexer.next();
23     }
24     return this.peeked;
25   }
26
27   take () {
28     const peeked = this.peeked;
29   }
30 }
31
```

antes



```
1 export default class PeekableLex {
2   constructor (lexer) {
3     this.peeked = null;
4     this.lexer = lexer;
5   }
6
7   wasPeeked () {
8     return this.peeked != null;
9   }
10
11   wasNotPeeked () {
12     return this.peeked == null;
13   }
14
15   isPeekTypeOf (type) {
16     const peeked = this.peek();
17     return peeked instanceof type;
18   }
19
20   peek () {
21     if (this.wasNotPeeked()) {
22       this.peeked = this.lexer.next();
23     }
24     return this.peeked;
25   }
26
27   take () {
28     const { peeked } = this.peeked;
29   }
30 }
31
```

depois