Qualidade e Teste

Integrantes:
Daniel Verginio Lima
Brendo Costa
Isaac Ferreira
Mathews

Ferramentas utilizadas

- Vitest
- Stryker
- Playwright
- ESLint

Descrição do Tabela-Verdade

- Gerador de tabela-verdade de expressões lógicas
- Código aberto
- JavaScript, React e Gatsby
- Composta por quatro módulos principais:
  - Source
  - Lexer
  - Parser
  - Generator

Requisitos Funcionais

Nº	Descrição
1	A aplicação deve ser capaz de receber uma sequência lógica de tamanho arbitrário como entrada pelo usuário.
2	A aplicação deve permitir que a linguagem da interface seja alterada.
3	A aplicação deve fornecer uma tabela-verdade de saída logicamente correta.
4	A aplicação deve aceitar as diferentes representações para um mesmo operador lógico.
5	A aplicação deve levar em conta a precedência dos operadores lógicos fornecidos na expressão de entrada.

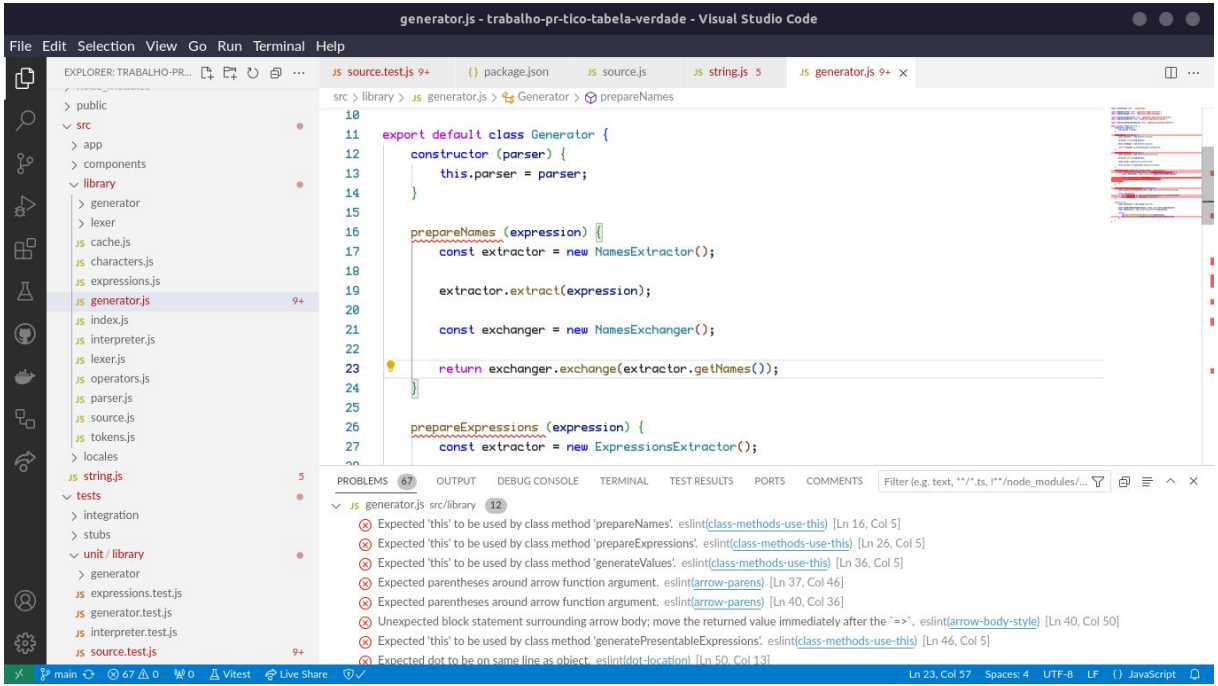
Requisitos Não Funcionais

Nº	Característica	Descrição
1	Eficiência de Desempenho	O processamento da sequência de entrada deve ser realizado de forma automática, sem intervenção do usuário.
2	Usabilidade	A interface de usuário deve se comportar de maneira responsiva e imediata a entradas incorretas do usuário.
3	Usabilidade	A interface de usuário deve fornecer ao usuário atalhos para formulação da expressão de entrada.
4	Portabilidade Adaptabilidade	A aplicação deve ser acessível através de dispositivos móveis sem perdas quanto a suas capacidades funcionais e não funcionais.
5	Manutenibilidade Modularidade	Os módulos da aplicação devem estar disponíveis sob a forma de componentes.
6	Manutenibilidade Analisabilidade	As dependências externas diretas da aplicação devem estar documentadas.

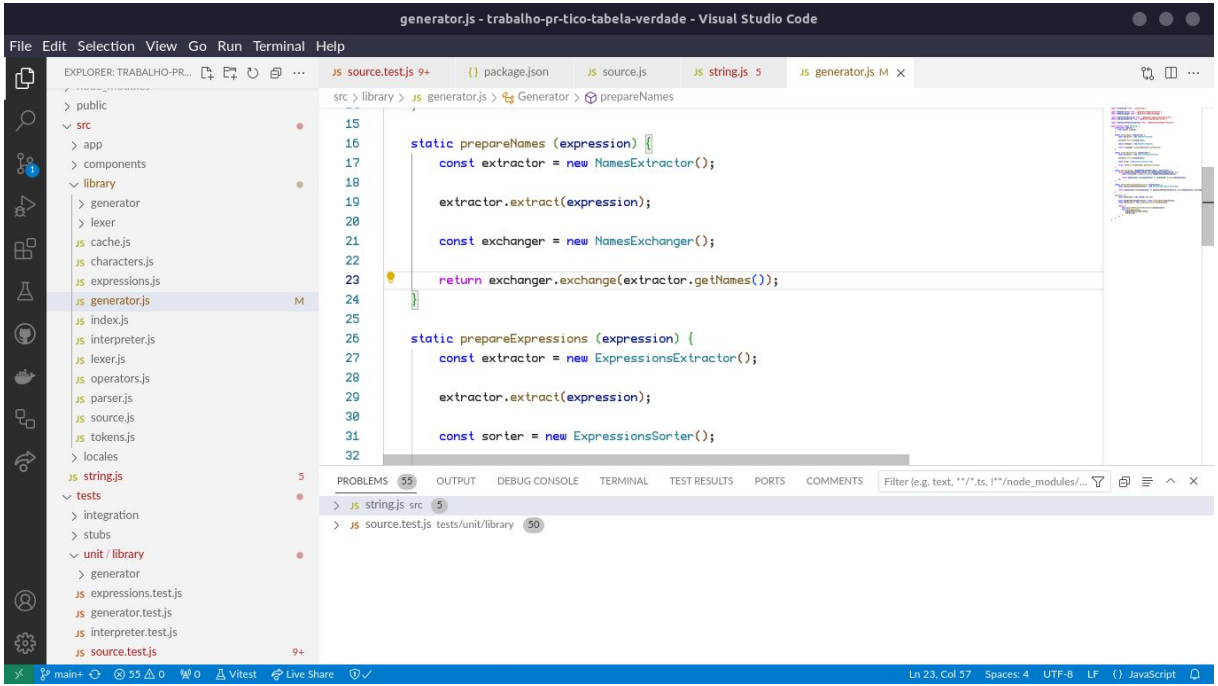
Requisitos Não Funcionais

7	Portabilidade Instalabilidade	Todo o processo de instalação da aplicação deve ser automatizado com a mínima intervenção do usuário.
8	Eficiência de Desempenho	A aplicação não deve ocupar mais que 100 megabytes de memória de acesso dinâmico quando em execução.
9	Segurança	As dependências externas da aplicação devem ser auditáveis quanto a vulnerabilidades de segurança.
10	Portabilidade Adaptabilidade	A aplicação deve ser portável para os navegadores *web* Google Chrome, Microsoft Edge, e Mozilla Firefox sem perdas quanto a suas capacidades funcionais e não funcionais.

Análise Estática



Análise Estática



Testes Unitários

- Definimos a menor unidade como sendo a classe;
- Verificam as funcionalidades de cada classe;
- Executados com o Vitest.

Testes Unitários

```
it('Should not generate table if expression is null', () => {
    const generator = new Generator({
        parse () {
            return null;
        }
    });

    expect(() => generator.generate())
        .toThrowError('Unexpected expression');
});

it('Check Next And()', () => {
    const lx = new Lexer({
        skipWhile(){},

        peek(){
            return '&'
        },

        bump(){}
    })
    const output = lx.next()
    expect(output).toBeInstanceOf(tokens.And);
});
```

```
it.each([
    ["兄弟 & 妹", "弟"],
    ["형제 | 자매", "제"],
    ["брат & !сестра", "р"],
    ["🍕🍫 -> 😁", "🍫"]
])('Should be able to bump Unicode characters', (input, expected) => {
    let source = new Source(input);
    source.bump();
    expect(source.peek()).toEqual(expected);
});

it('Should call bump after skipWhile with a function retur

    const source = new Source('wwwxyz');
    const spy = vi.spyOn(source, 'bump');
    source.skipWhile(ch => {
        return ch === 'w';
    });
    expect(spy).toHaveBeenCalledTimes(3);

});
```

Testes de Integração

- Abordagem Big Bang
  - Projeto pequeno
  - Todas as unidades já estavam disponíveis no momento do começo dos testes
- Realizamos o uso de mocks e stubs para verificar casos hipotéticos.

Testes de Integração

```
import {
    Expression,
    UnaryExpression,
    BinaryExpression,
} from '../../src/library/expressions';

export class BadExpression extends Expression {}
export class BadUnaryExpression extends UnaryExpression {}
export class BadBinaryExpression extends BinaryExpression {}
```

```
it('Should support iteration over ASCII and Unicode character', () => {
    let source = new Source("ab 🍮cde🍕fghi🍫jk🍰lnmopqrstuvwxyzブラジル");
    let iterator = source.input[Symbol.iterator]();
    const spy = vi.spyOn(iterator, 'next');
    for (let character of iterator) {}
    expect(spy).toHaveBeenCalledTimes(36); // 35 characters + 1 EOF
});
```

```
it('Should consume alphanumeric characters as name toke
    const lexer = createLexerFromString('abcXYZ123');

    expect(lexer.next()).toBeInstanceOf(Name);
});
```

```
it('Should generate values for single character varia
    const generator = createGeneratorFromString('a')

    expect(generator.generate()).toEqual([
        ['a'],
        [[true], [false]]
    ]);
});
```

Testes de Sistema

- Verificam o requisito funcional da corretude na geração das tabelas, validação de entradas do usuário e alteração do idioma da interface, bem como os requisitos não-funcionais de usabilidade como atalhos para operações, responsividade dos elementos da interface e portabilidade entre diferentes navegadores.

Testes de Sistema

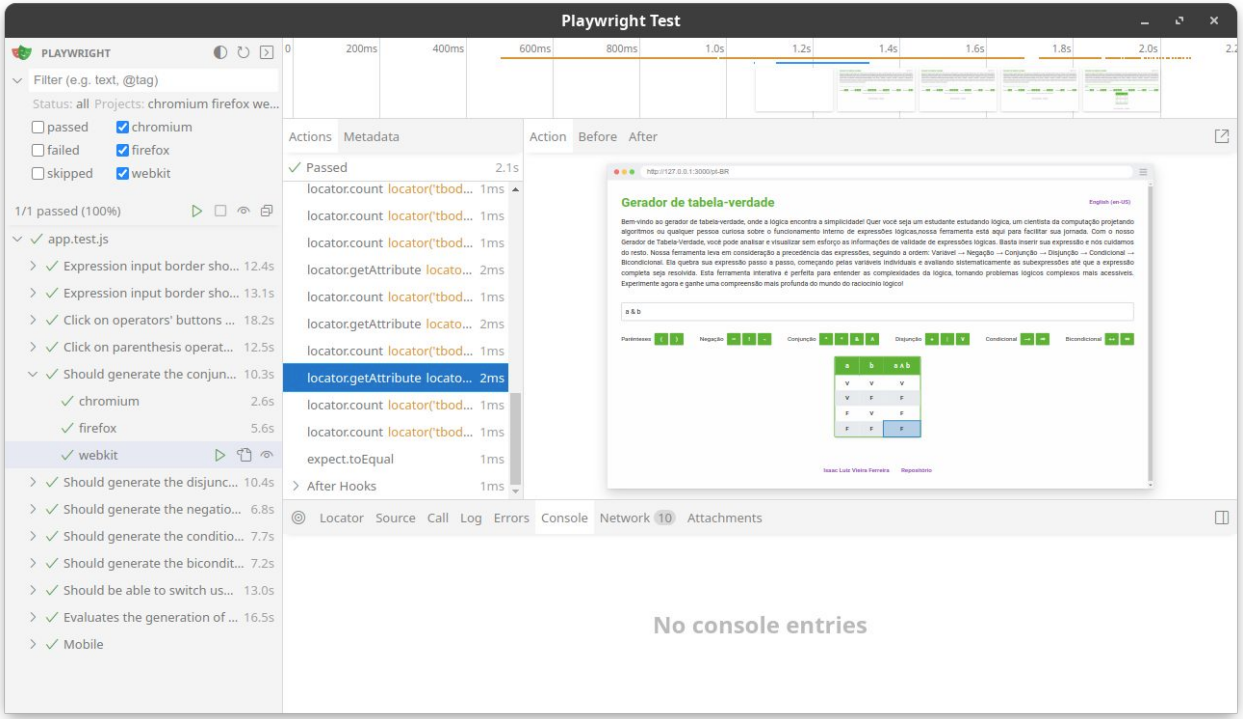
```
test('Expression input border should turn to primary color when the user enter a valid input values', async ({ page }) => {
    const validInputCases = [
        '                  a & b',
        '          a &    b          ',
        'a           & b',
        'a & b',
        '(a & b)',
        '(a & b) & c',
        '(a | b) -> c',
        '(a ^ b) -> (c)'
    ];
    for (const testCase of validInputCases) {
        const input = page.getByRole('textbox');
        await input.fill(testCase);
        await expect(input).toHaveClass(/border-primary/);
    }
});
```

Testes de Sistema

```
test('Should generate the biconditional truth table', async ({ page }) => {
    const input = page.getByRole('textbox');
    await input.fill('a <-> b');

    await expect(await getTableData(page)).toEqual([
        [true, true, true],
        [true, false, false],
        [false, true, false],
        [false, false, true],
    ]);
});
```

Testes de Sistema



Técnicas de Teste Funcionais

- Utilizamos o particionamento em classes de equivalência para testar o comportamento da implementação diante de conjuntos possíveis de entrada do usuário.

Técnicas de Teste Funcionais

```
it.each([
    ["兄弟 & 妹", "弟"],
    ["형제 | 자매", "제"],
    ["брат & !сестра", "р"],
    ["🍕🍫 -> 😁", "🍫"]
])('Should be able to bump Unicode characters', (input, expected) => {
    let source = new Source(input);
    source.bump();
    expect(source.peek()).toEqual(expected);
});
```

```
test('Expression input border should become red when the user enter a invalid input values', async ({ page }) => {
    const invalidInputsCases = [
        '🍎 & 🍌',
        '白 & 黒',
        '123 & 456',
        'a b &',
        '5',
        '🍎',
        ' '
    ];
    for (const testCase of invalidInputsCases) {
        const input = page.getByRole('textbox');
        await input.fill(testCase);
        await expect(input).toHaveClass(/border-red-500/);
    }
});
```

Técnicas de Teste Estruturais

- Aplicamos o critério todas-arestas visando a obter uma cobertura superior a 80%;
- Análise da cobertura realizada através do Vitest.

Técnicas de Teste Estruturais

All files

100% Statements `1002/1002` **100%** Branches `255/255` **100%** Functions `93/93` **100%** Lines `1002/1002`

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
src		100%	25/25	100%	2/2	100%	2/2	100%	25/25
src/library		100%	606/606	100%	159/159	100%	49/49	100%	606/606
src/library/generator/expressions		100%	58/58	100%	16/16	100%	7/7	100%	58/58
src/library/generator/names		100%	90/90	100%	24/24	100%	8/8	100%	90/90
src/library/generator/presenter		100%	178/178	100%	44/44	100%	20/20	100%	178/178
src/library/lexer		100%	45/45	100%	10/10	100%	7/7	100%	45/45

Técnicas de Teste Estruturais

All files

100% Statements 997/997 **100%** Branches 257/257 **100%** Functions 91/91 **100%** Lines 997/997

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
src		100%	25/25	100%	2/2	100%	2/2	100%	25/25
src/library		100%	601/601	100%	158/158	100%	47/47	100%	601/601
src/library/generator/expressions		100%	58/58	100%	16/16	100%	7/7	100%	58/58
src/library/generator/names		100%	90/90	100%	24/24	100%	8/8	100%	90/90
src/library/generator/presenter		100%	178/178	100%	47/47	100%	20/20	100%	178/178
src/library/lexer		100%	45/45	100%	10/10	100%	7/7	100%	45/45

Técnicas de Teste Baseadas em Defeitos

- Realizamos os testes de mutação através da ferramenta Stryker visando obter um *score* de mutação superior a 80%.

Técnicas de Teste Baseadas em Defeitos

File / Directory	Mutation score	Killed	Survived	Timeout	No coverage	Ignored	Runtime errors	Compile errors	Detected	Undetected	Total
All files	93.57	484	18	11	16	0	0	0	495	34	529
generator	94.51	153	9	2	0	0	0	0	155	9	164
lexer/peekable.js	100.00	20	0	0	0	0	0	0	20	0	20
cache.js	0.00	0	0	0	6	0	0	0	0	6	6
characters.js	100.00	16	0	0	0	0	0	0	16	0	16
expressions.js	100.00	43	0	0	0	0	0	0	43	0	43
generator.js	100.00	12	0	0	0	0	0	0	12	0	12
index.js	0.00	0	0	0	1	0	0	0	0	1	1
interpreter.js	100.00	49	0	0	0	0	0	0	49	0	49
lexer.js	93.38	123	9	4	0	0	0	0	127	9	136
operators.js	0.00	0	0	0	9	0	0	0	0	9	9
parser.js	100.00	57	0	3	0	0	0	0	60	0	60
source.js	100.00	9	0	2	0	0	0	0	11	0	11
tokens.js	100.00	2	0	0	0	0	0	0	2	0	2

Técnicas de Teste Baseadas em Defeitos

File / Directory	Mutation score	Killed	Survived	Timeout	No coverage	Ignored	Runtime errors	Compile errors	Detected	Undetected	Total
All files	96.73	493	1	10	16	0	0	0	**503**	**17**	**520**
generator	99.37	156	1	2	0	0	0	0	**158**	**1**	**159**
lexer/peekable.js	100.00	20	0	0	0	0	0	0	**20**	**0**	**20**
cache.js	0.00	0	0	0	6	0	0	0	**0**	**6**	**6**
characters.js	100.00	16	0	0	0	0	0	0	**16**	**0**	**16**
expressions.js	100.00	43	0	0	0	0	0	0	**43**	**0**	**43**
generator.js	100.00	12	0	0	0	0	0	0	**12**	**0**	**12**
index.js	0.00	0	0	0	1	0	0	0	**0**	**1**	**1**
interpreter.js	100.00	49	0	0	0	0	0	0	**49**	**0**	**49**
lexer.js	100.00	128	0	4	0	0	0	0	**132**	**0**	**132**
operators.js	0.00	0	0	0	9	0	0	0	**0**	**9**	**9**
parser.js	100.00	58	0	2	0	0	0	0	**60**	**0**	**60**
source.js	100.00	9	0	2	0	0	0	0	**11**	**0**	**11**
tokens.js	100.00	2	0	0	0	0	0	0	**2**	**0**	**2**