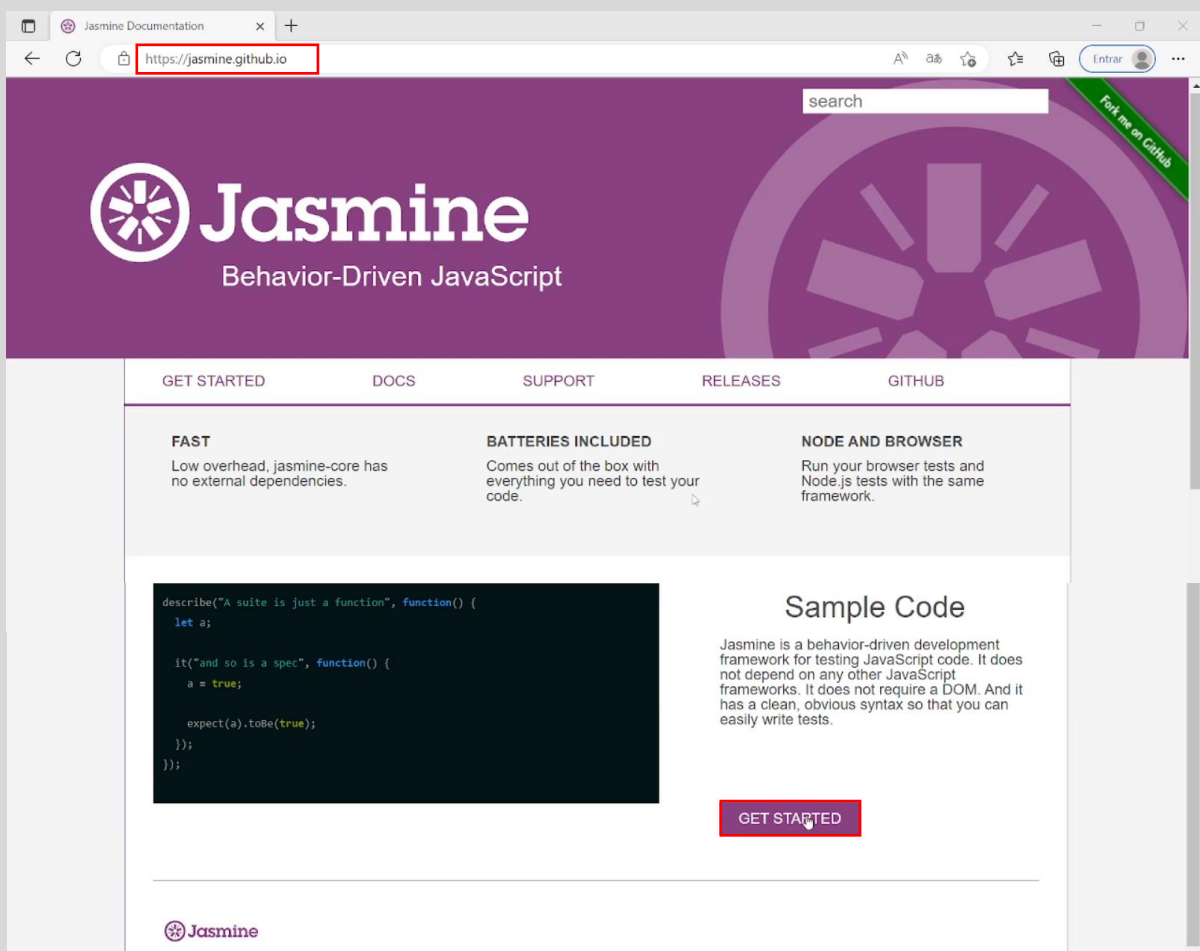


Atividade

Nessa atividade, você vai fazer um teste unitário de código Javascript, utilizando o Jasmine.

1. Acesse o endereço **<https://jasmine.github.io/>** e clique em **Get Started**.



The screenshot shows a web browser window with the address bar displaying <https://jasmine.github.io/>. The page features the Jasmine logo and the text "Behavior-Driven JavaScript". A navigation bar includes links for GET STARTED, DOCS, SUPPORT, RELEASES, and GITHUB. The main content area highlights three key features: FAST (Low overhead, jasmine-core has no external dependencies), BATTERIES INCLUDED (Comes out of the box with everything you need to test your code), and NODE AND BROWSER (Run your browser tests and Node.js tests with the same framework). A "Sample Code" section displays a JavaScript snippet for a unit test, followed by a "GET STARTED" button.

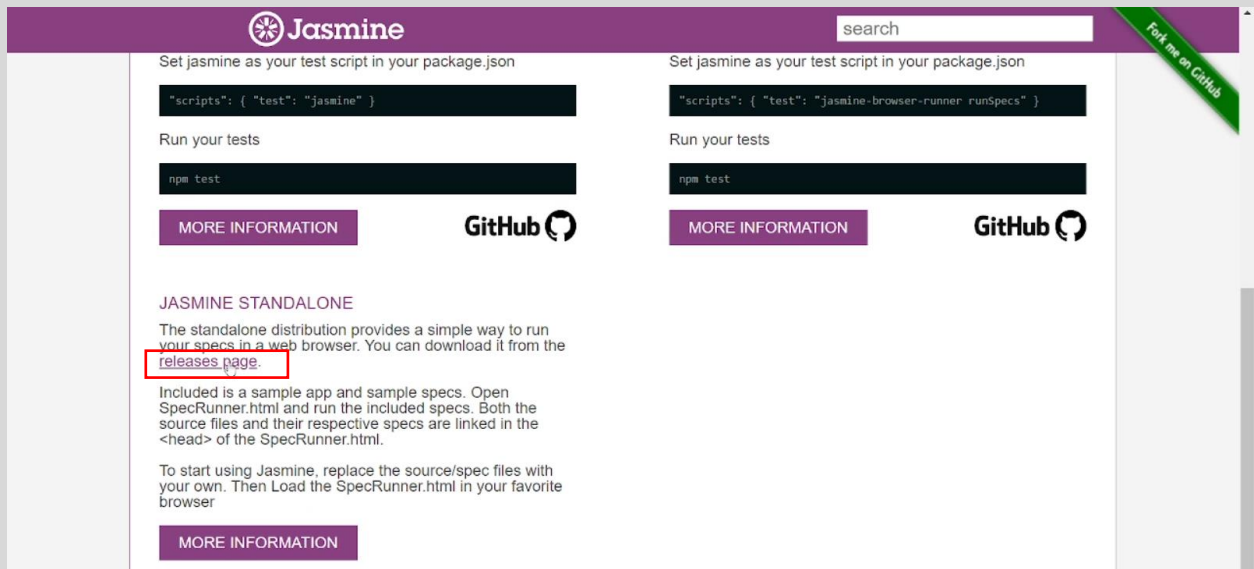
```
describe("A suite is just a function", function() {  
  let a;  
  
  it("and so is a spec", function() {  
    a = true;  
  
    expect(a).toBe(true);  
  });  
});
```

Sample Code

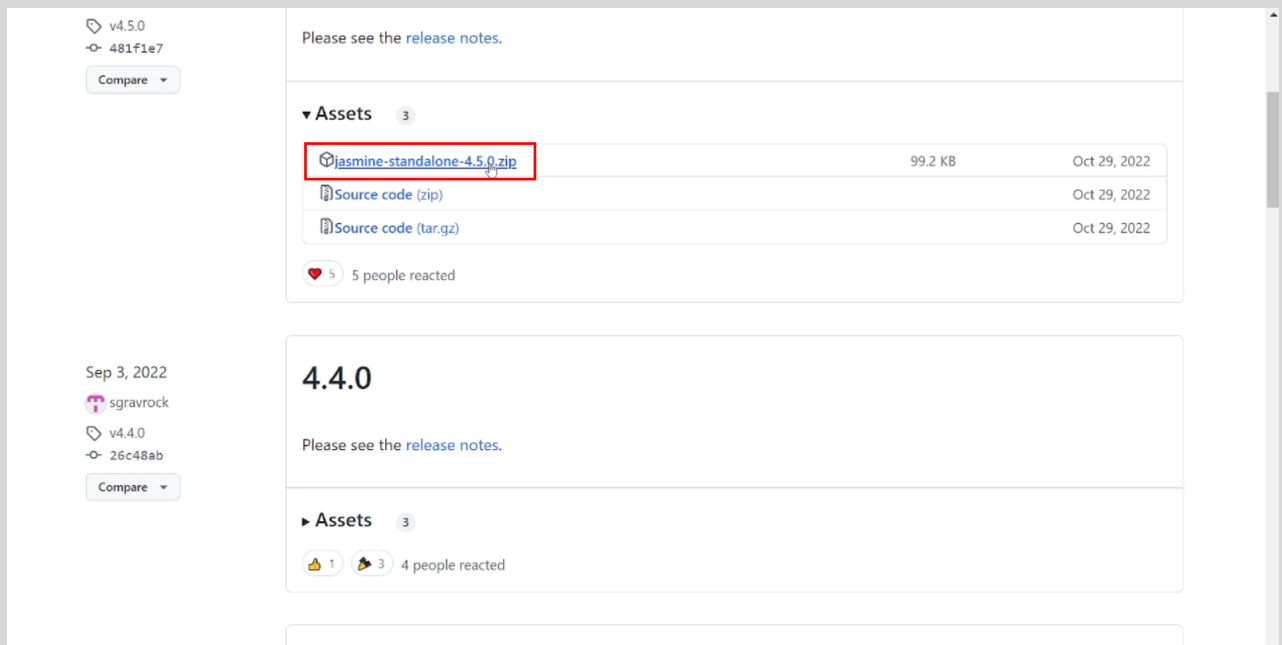
Jasmine is a behavior-driven development framework for testing JavaScript code. It does not depend on any other JavaScript frameworks. It does not require a DOM. And it has a clean, obvious syntax so that you can easily write tests.

[GET STARTED](#)

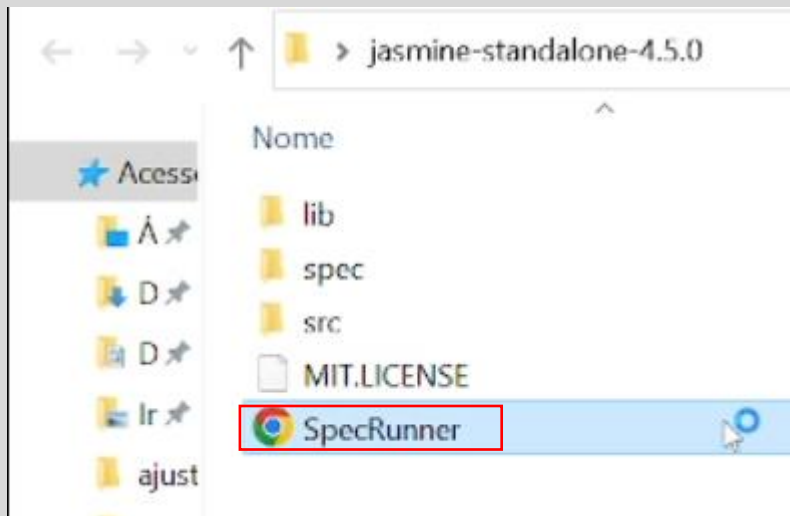
2. Na janela seguinte, role a tela até encontrar **Jasmine Standalone** e clique em **releases page**.



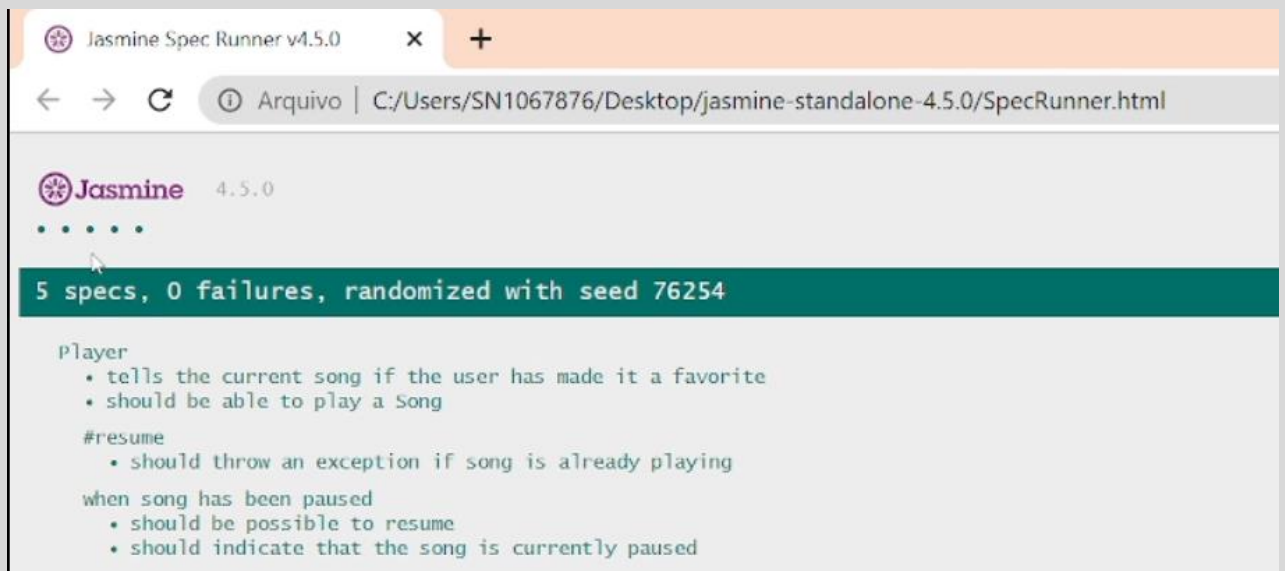
3. Clique no link **jasmine-standalone** para baixar o arquivo compactado.



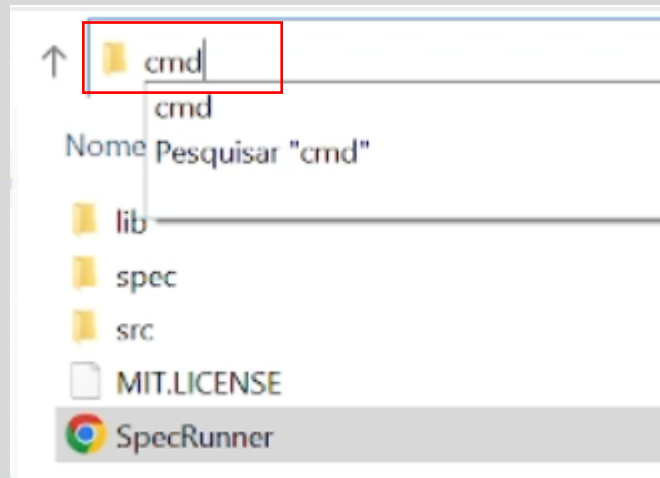
4. Descompacte o arquivo salvo, abra a pasta e clique duas vezes em **SpecRunner** para executar o Jasmine.



5. O Jasmine será executado no navegador.



6. Retorne para a pasta do **jasmine-standalone** e digite **cmd** na barra de localização e dê **enter**.

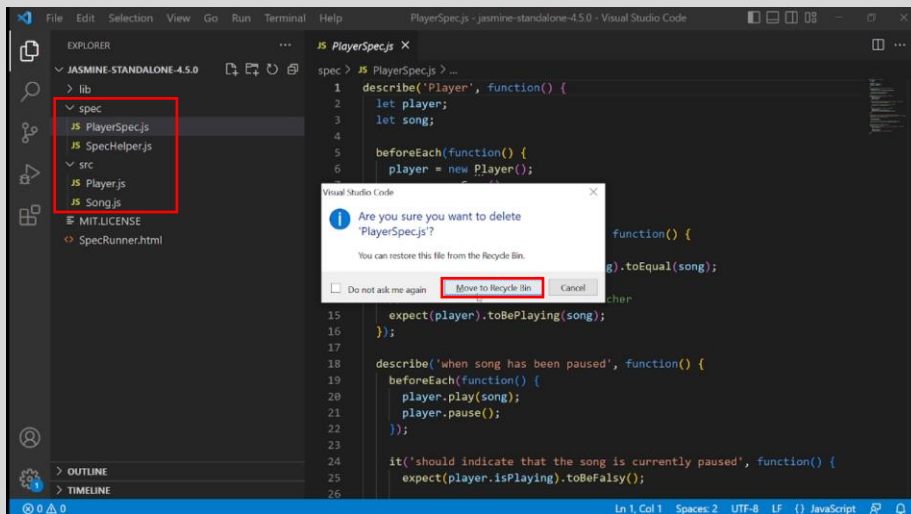


7. No terminal, digite **code .** para abrir o VS Code.

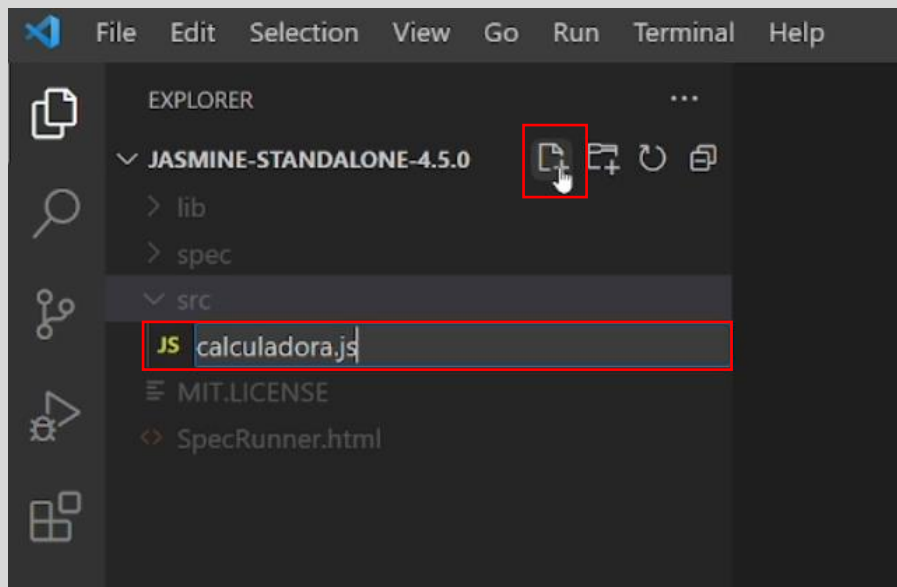
```
C:\Windows\System32\cmd.exe
Microsoft Windows [versão 10.0.19045.2364]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\SN1067876\Desktop\jasmine-standalone-4.5.0>code .
```

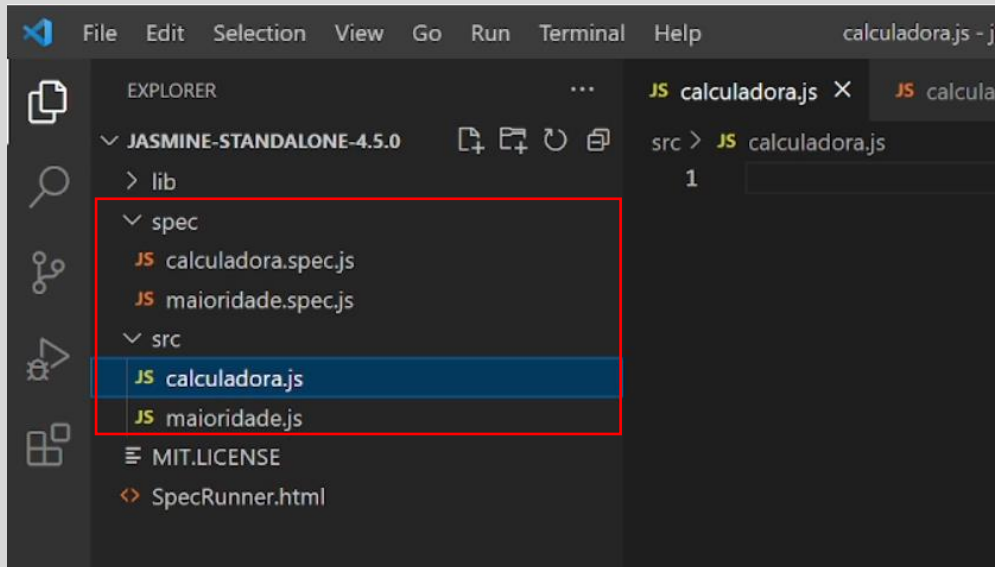
8. No VS Code, apague os arquivos teste **PlayerSpec.js**, **SpecHelper.js**, **Player.js** e **Song.js**. Mantenha as pastas **src** e **spec**. Para isso, selecione os arquivos, clique com o botão direito e selecione **delete**. Na janela de alerta, selecione **Move to Recycle Bin**.



9. Selecione a pasta **src** e clique no ícone **Novo Arquivo**. Nomeie o arquivo como **calculadora.js**.



10. Crie também o arquivo **maioridade.js** na pasta **src**, além dos arquivos **calculadora.spec.js** e **maioridade.spec.js** na pasta **spec**.



11. A seguir, implemente o algoritmo da calculadora no arquivo **calculadora.js**, conforme o código a seguir.

```
//Função de soma

const soma = function(num1,num2){
    return(num1 + num2);
}

const subtracao = function(num1,num2){
    return(num1 - num2);
}

const multiplicacao = function(num1,num2){
    return(num1 * num2);
}

const divisao = function(num1,num2){
    return(num1 / num2);
}
```

12. Agora, implemente o algoritmo para calcular a maioria no arquivo **maioridade.js**, conforme o código a seguir.

```
const maioria = function(idade){

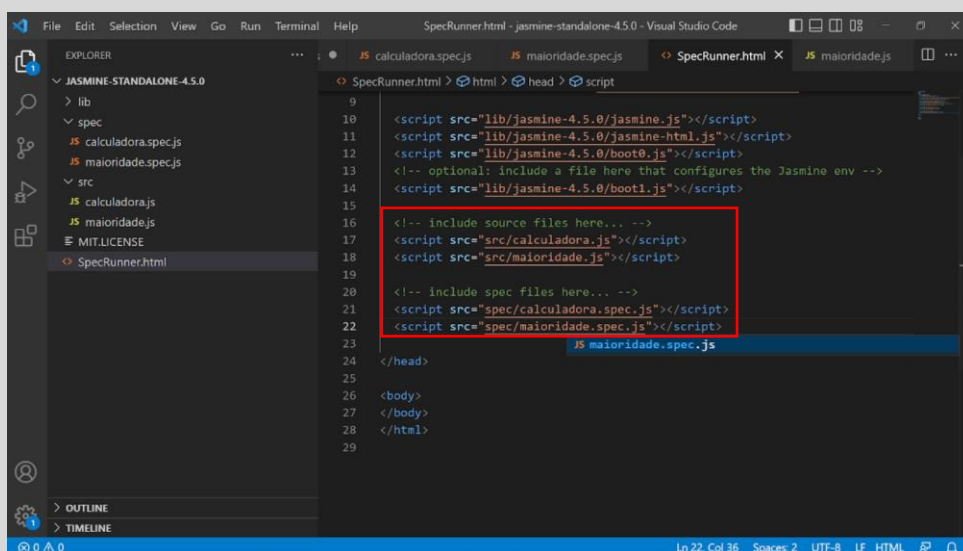
    if(idade <=0){
        return 'Idade Inválida';
    }

    if(idade>=18){
        return 'Maior Idade';
    }else{
        return 'Menor Idade';
    }
}
```

13. Dentro de **SpecRunner.html**, atualize o nome dos arquivos de referência, conforme código a seguir.

```
<!-- include source files here... -->
<script src="src/calculadora.js"></script>
<script src="src/maioridade.js"></script>

<!-- include spec files here... -->
<script src="spec/calculadora.spec.js"></script>
<script src="spec/maioridade.spec.js"></script>
```

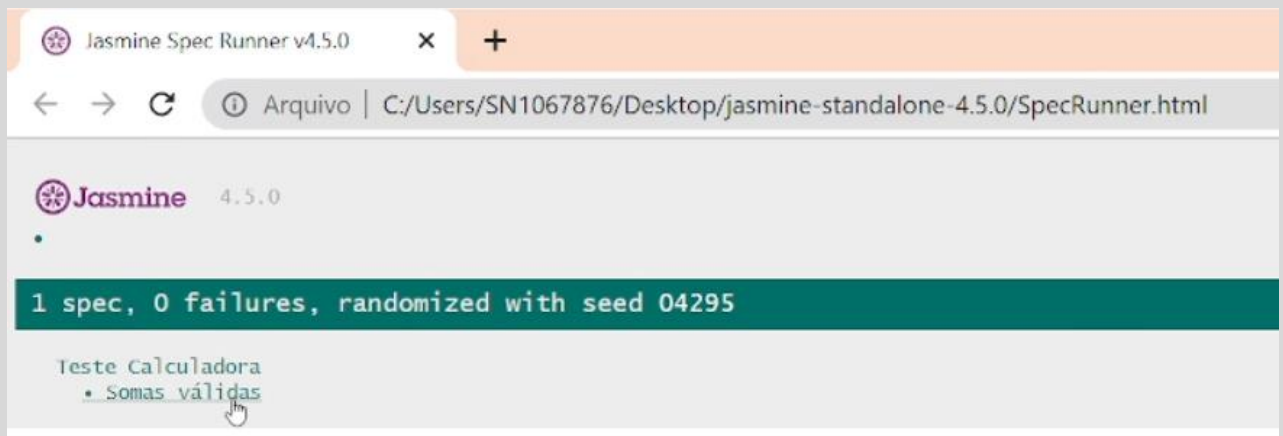


14. No arquivo **calculadora.spec.js**, crie o algoritmo para verificar operações aritméticas, como:

```
describe("Teste Calculadora", function(){  
    it("Somas válidas", function(){  
        expect(soma(1,2)).toBe(3);  
        expect(soma(9,9)).toBe(18);  
    });  
});
```

Nesse trecho, esperamos que a soma $1 + 2 = 3$ e que a soma $9 + 9 = 18$. Salve as alterações em seus arquivos.

15. Execute o **SpecRunner.html** em seu navegador, clicando duas vezes sobre o arquivo na pasta do projeto. O navegador deve mostrar um resultado como a seguir:

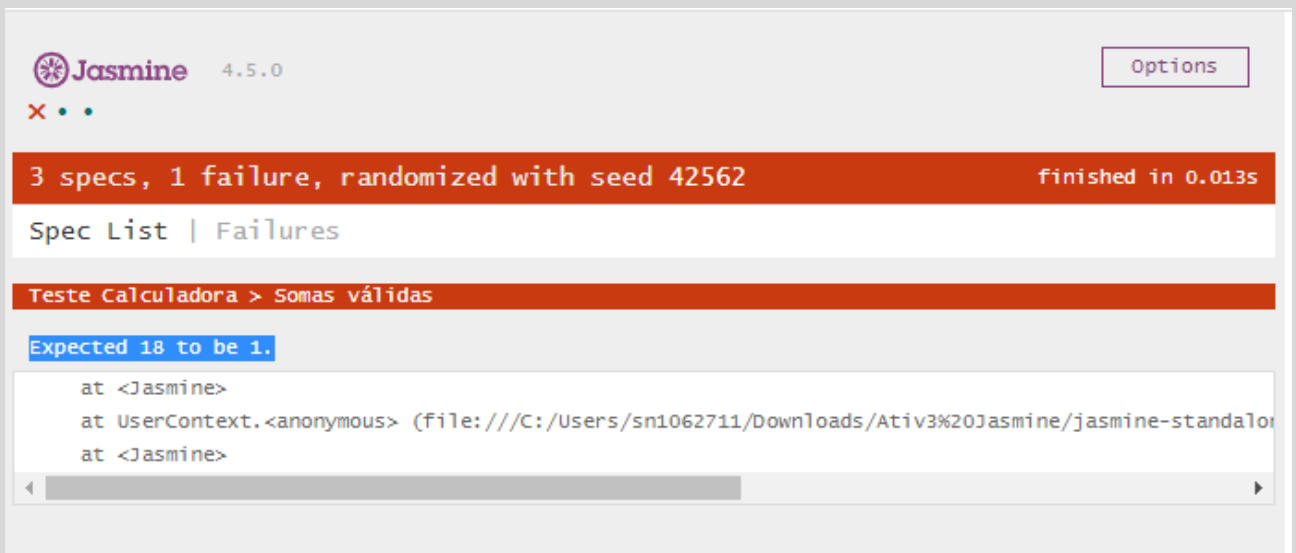


16. Em **calculadora.spec.js**, mude o valor da segunda soma para 1:

```
describe("Teste Calculadora", function(){  
  it("Somas válidas", function(){  
    expect(soma(1,2)).toBe(3);  
    expect(soma(9,9)).toBe(1);  
  });  
});
```

Nesse trecho, esperamos que a soma $1 + 2 = 3$ e que a soma $9 + 9 = 1$. Salve as alterações em seus arquivos.

17. Atualize o **SpecRunner.html** em seu navegador e o resultado deve ser como a seguir:



Note que o teste mostra qual o erro detectado.

18. No arquivo **maioridade.spec.js**, implemente o teste de verificação de idade.

```
describe("Teste de validação - Maior Idade", function(){
    it("Validação Maior idade", function(){
        expect(maioridade(18)).toBe('Maior Idade');
    });
});

describe("Teste de validação - Menor de Idade", function(){
    it("Validação Menor Idade", function(){
        expect(maioridade(10)).toBe('Menor Idade');
    });
});
```

Nesse trecho, esperamos que 18 seja 'Maior Idade' e que 10 seja 'Menor Idade'. Salve as alterações.

19. Atualize o **SpecRunner.html** em seu navegador e o resultado deve ser como a seguir:



18. Em **maioridade.spec.js**, mude o valor da segunda idade para 20:

```
describe("Teste de validação - Maior Idade", function(){  
    it("Validação Maior idade", function(){  
        expect(maioridade(18)).toBe('Maior Idade');  
    });  
});  
  
describe("Teste de validação - Menor de Idade", function(){  
    it("Validação Menor Idade", function(){  
        expect(maioridade(20)).toBe('Menor Idade');  
    });  
});
```

Nesse trecho, esperamos que 20 seja 'Menor Idade'. Salve as alterações.

19. Atualize o SpecRunner.html em seu navegador e o resultado deve ser como a seguir:



Note que o teste mostra qual o erro detectado.

Documentação

1. Baixe o modelo de plano de teste e preencha os dados solicitados para o teste unitário de sistema, tanto para a calculadora quanto para a maioria.

Título do Plano de Testes: Plano de Testes Unitário o Sistema De Calculadora

Data de Criação: 20/01/2023

Autor: Felipe

Objetivo: Este plano de testes tem como objetivo definir as estratégias, processos e recursos necessários para a realização de testes unitários do Sistema de Calculadora.

Escopo: O escopo deste plano de testes inclui a verificação da funcionalidade de todas as funcionalidades do sistema X.

Ambiente de Teste:

- Sistema Operacional: Windows 10
- Banco de Dados: MySQL

Salve as alterações no doc e envie em formato pdf.

Dica!

Para acessar informações sobre o sistema operacional e hardware de sua máquina, use o atalho **Windows + pause**.

