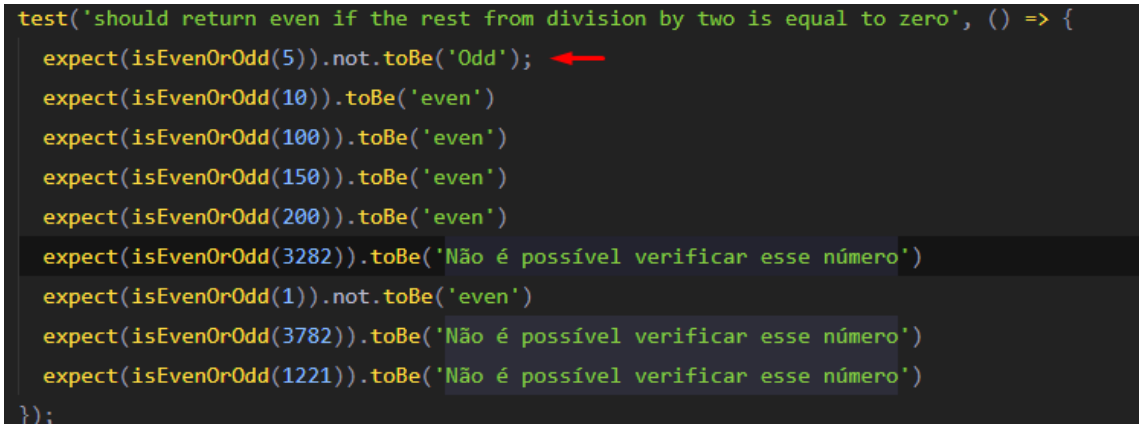
Documentação do teste Probrain

Objetivo: O objetivo do teste foi criar erros e debugalos.

Requisitos do teste: O teste foi feito no visual studio code, na linguagem de Javascript e seguindo um padrão de metodologia TDD (Test Driven Development). Eu utilizei também o recurso de uma biblioteca chamada Jest para o auxílio do teste.

Implementação 1

Fiz três blocos de testes, um contendo verificação de números pares, o segundo contendo a verificação de números impares e o terceiro contendo um loop verificando se é impar ou par, de número zero a mil. Algumas vezes tive problemas com grafia, iniciando com palavras maiúsculas. Fiz verificações utilizando a propriedade.not nos métodos de expect.



Acima na imagem está um dos erros de grafia que cometi enquanto estava fazendo os testes. O valor é pra ser escrito em letra minúscula, e eu estava pondo maiúscula.

Implementação 2

Fiz novamente três blocos de testes, neles eu coloquei verificações de números primos. Se o resultado da divisão for menor que o divisor, o número é primo e saia 'true', caso não fosse, sairia 'false'. Implementei também uma verificação de números maiores do que mil. Caso fosse maior, a função reportaria 'Não é possível verificar esse

número'. Nessa segunda implementação, não tive tanta dificuldade nem tantos erros, já que a primeira foi mais desafiadora para mim.

```
test('should validate numbers greater than one thousand', () => {
  expect(isPrime(1001)).toBe('Não é possível verificar esse número')
  expect(isPrime(241)).toBe(true)
  expect(isPrime(3815)).toBe('Não é possível verificar esse número')
  expect(isPrime(67)).not.toBe(false)
  expect(isPrime(87512451)).toBe('Não é possível verificar esse número')
  expect(isPrime(21)).toBe(false)
  expect(isPrime(999)).toBe(false)
  expect(isPrime(81511)).toBe('Não é possível verificar esse número')
})
```

Caso não tivesse a string 'Não é possível verificar esse número' ou se ela estivesse escrita de maneira errada, o Jest reportava um erro. Então, em valores acima de mil, só poderia ser reportado o erro que criei com essa mensagem, caso não fosse dava 'false' e bugava o código.

Implementação 3

Verificando se o número passado como argumento para função 'hasValueInFibonacciSequence' está presente na sequência de Fibonacci, com limite máximo de valor até 1000. Também não tive tantos erros.

Implementação 4

Mostrando a sequência de múltiplos de quatro e adicionando nelas a palavra 'pin'. Não tive erros.

Conclusão: Com o passar dos desafios, os testes foram ficando cada vez mais fácil e a lógica foi ficando mais clara. Não senti dificuldade em fazer os testes apesar de não ter costume com QA, mas achei interessante a forma de trabalho e pesquisando na internet utilizei uma metodologia chamada TDD. Nele eu comecei a testar antes de fazer a função, começando com testes básicos de grafia, depois fui fazendo os testes de funcionalidade com o auxílio da biblioteca Jest.