Explicação do código

1. Cancellable

   A função executeWithDelay foi criada para permitir a execução de uma função com atraso, dando a opção de cancelamento antes do término desse atraso.

```
const cancellable = function(fn, args, t) {
    let timeoutId;
    let cancelled = false;

    function cancelFn() {
        cancelled = true;
        clearTimeout(timeoutId);
    }

    function delayExecution() {
        if (!cancelled) {
            fn.apply(null, args);
        }
    }

    timeoutId = setTimeout(delayExecution, t);

    return cancelFn;
};

module.exports = cancellable;
```

   Detalhes da Implementação:

   timeoutId: Identificador do setTimeout usado para agendar a execução.
   cancelled: Variável para rastrear se a função foi cancelada.
   cancelFn(): Função de cancelamento que define cancelled como true e cancela o setTimeout.
   delayExecution(): Função executada após o atraso, que verifica se a função foi cancelada antes da execução.
   return cancelFn: A função executeWithDelay retorna cancelFn, permitindo que os usuários cancelem a execução antes do atraso.

2. Testes em Jest

Os testes foram implementados usando o framework Jest para garantir que a função *cancellable* funcione conforme esperado em diferentes cenários. Aqui estão alguns exemplos:

```
const executeWithDelay = require('./cancellable');

// Teste 1: Verifica se a função é executada após o atraso
test('Executa a função após o atraso especificado', (done) => {
  const mockFn = jest.fn();
  const cancelFunction = executeWithDelay(mockFn, [], 100);

  setTimeout(() => {
    expect(mockFn).toHaveBeenCalled();
    done();
  }, 200); // Aguarda tempo suficiente para garantir a execução da
função
});

// Teste 2: Verifica se a função é cancelada corretamente
test('Cancela a execução da função', (done) => {
  const mockFn = jest.fn();
  const cancelFunction = executeWithDelay(mockFn, [], 100);

  // Chama a função de cancelamento antes do atraso
  cancelFunction();

  setTimeout(() => {
    expect(mockFn).not.toHaveBeenCalled();
    done();
  }, 200); // Aguarda tempo suficiente para garantir que a função não
foi executada
});
```

Detalhes dos Testes:

Teste 1: Garante que a função seja executada após o atraso especificado.
Teste 2: Confirma que a função é cancelada corretamente quando a função de cancelamento é chamada antes do atraso.

Esses testes fornecem uma cobertura básica para validar o comportamento da função em diferentes situações.