



**PUC Minas**

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS  
GERAIS ENGENHARIA DE SOFTWARE  
TESTE DE SOFTWARE**

Beatriz Soares de Melo  
Belle Nerissa Aguiar Elizeu  
Júlia Leite de Souza  
Kesley Alexsandro Vieira Severino  
Letícia Amanda Franco Gonçalves  
Octávio Tabai Ribeiro Lage

**TRABALHO PRÁTICO 2**

**Jest**

Trabalho de Pesquisa desenvolvido para a  
disciplina Teste de Software

Professor: Johnatan Alves de Oliveira

Belo Horizonte

2022

## SUMÁRIO

<b>1 DESCRIÇÃO DO SOFTWARE</b>	<b>3</b>
<b>2 CASOS DE TESTE REALIZADOS</b>	<b>4</b>
<b>3 CÁLCULO DE COBERTURA DE TESTES (COVERAGE)</b>	<b>4</b>
<b>4 SOFTWARE UTILIZADO PARA COMPUTAR A COBERTURA DE TESTES</b>	<b>5</b>
<b>5 RESULTADOS</b>	<b>5</b>

## 1 DESCRIÇÃO DO SOFTWARE

O software construído para a realização da computação de cobertura de testes é um sorteador de amigo secreto. Na primeira etapa é pedido do usuário a inserção dos nomes de cada um dos integrantes que são mostrados em uma lista sequencial.



Figura 01 — Interface inicial da aplicação de sorteador

Após o clique do botão “Iniciar brincadeira” é possível selecionar o integrante que irá sortear seu amigo secreto, e após o clique do sorteio é exibido o nome do integrante sorteado.

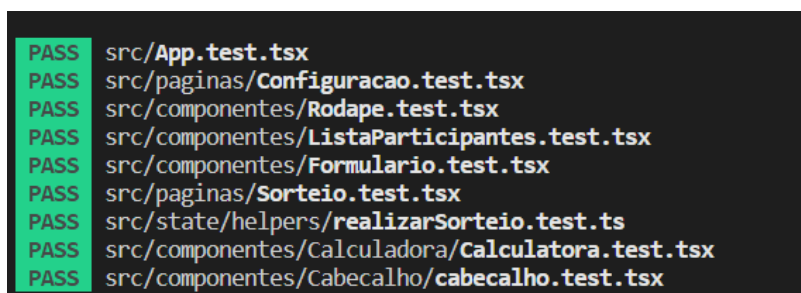


Figura 02 — Interface final da aplicação de sorteador

O seguinte software contém 17 arquivos e 20 métodos se considerar os arquivos responsáveis pela lógica da aplicação e retirar os demais responsáveis por estilização, a realização dos próprios testes, configurações da biblioteca Jest, da framework React, gerenciador de pacotes, entre outros.

## 2 CASOS DE TESTE REALIZADOS

No total foram realizados 16 casos de teste em toda a aplicação, totalizando 9 conjuntos de testes no qual abrange cada um único arquivo em específico como descrito na próxima imagem. Além disso, foram desenvolvidos mais 3 casos de testes do tipo Snapshot, que é um caso de teste específico que realiza testes de similaridade entre os componentes de Interface de usuário (UI) com o renderizado no arquivo de teste.



```
PASS src/App.test.tsx
PASS src/paginas/Configuracao.test.tsx
PASS src/componentes/Rodape.test.tsx
PASS src/componentes/ListaParticipantes.test.tsx
PASS src/componentes/Formulario.test.tsx
PASS src/paginas/Sorteio.test.tsx
PASS src/state/helpers/realizarSorteio.test.ts
PASS src/componentes/Calculadora/Calculadora.test.tsx
PASS src/componentes/Cabecalho/cabecalho.test.tsx
```

Figura 03 — Log do console dos locais de cada conjunto de teste

## 3 CÁLCULO DE COBERTURA DE TESTES (COVERAGE)

O cálculo realizado pela biblioteca tem diferentes abordagens, totalizando quatro tipos de resultados gerados para cada caso de teste, que são: Cálculo por declaração (statement), por ramificação (branch), por função (function) e por total de linhas (total lines).

Os testes por declaração abordam o total de cobertura de teste por declaração executadas pelo programa, os de ramificação abordam todos os possíveis caminhos de cada execução do programa, como loops, exceções, terminais de switches entre outros, os de função abordam a cobertura de testes por funções existentes em cada arquivo e por fim os testes por total de linhas abordam a cobertura contabilizando o total de linhas de cada arquivo.

## 4 SOFTWARE UTILIZADO PARA COMPUTAR A COBERTURA DE TESTES

Foi utilizado a própria framework Jest para realizar o cálculo de cobertura dos testes do sistema. A cobertura de código no Jest pode ser usada pelo comando `jest --coverage` ou `yarn test --coverage` e não requer pacotes ou configurações adicionais.

## 5 RESULTADOS

Os resultados obtidos com a cobertura dos testes mostram que obtivemos um total de 93,67 % de cobertura do sistema por casos de teste. Com mais detalhes na imagem abaixo é possível observar com mais detalhes a cobertura e seus totais de cobertura de cada arquivo da aplicação.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	93.82	90	88.46	93.67	
src	0	100	0	0	
App.tsx	0	100	0	0	8
index.tsx	0	100	100	0	7
src/componentes	100	100	100	100	
Formulario.tsx	100	100	100	100	
ListaParticipantes.tsx	100	100	100	100	
Rodape.tsx	100	100	100	100	
src/componentes/Cabecalho	100	100	100	100	
index.tsx	100	100	100	100	
src/componentes/Calculadora	100	100	100	100	
Calculadora.tsx	100	100	100	100	
src/componentes/Card	100	100	100	100	
index.tsx	100	100	100	100	
src/paginas	100	75	100	100	
Configuracao.tsx	100	100	100	100	
Sorteio.tsx	100	75	100	100	20
src/state	100	100	100	100	
atom.ts	100	100	100	100	
src/state/helpers	100	100	100	100	
realizarSorteio.ts	100	100	100	100	
src/state/hook	88	100	77.77	87.5	
useAdicionarParticipante.ts	100	100	100	100	
useListaDeParticipantes.ts	100	100	100	100	
useMensagemDeErro.ts	100	100	100	100	
useResultadoSorteio.ts	50	100	0	50	5
useSorteador.ts	66.66	100	50	66.66	10-11
Test Suites: 9 passed, 9 total					
Tests: 16 passed, 16 total					
Snapshots: 3 passed, 3 total					
Time: 4.712 s					
Ran all test suites.					