

Qualidade de Software

Tales Santos

Qualidade de Software

Aula 2

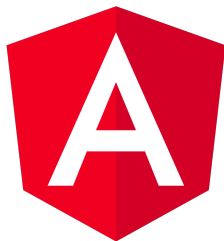
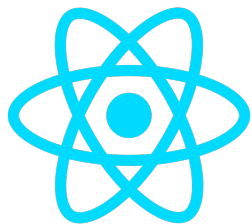
- Conhecer um pouco mais a ferramenta Jest
- Conhecer testes com dublês e espião
- Implementar testes de integração em aplicação real
- Implementar testes de unidade com dublês e espiões

Introdução ao Jest



Qualidade de Software

- Para teste de aplicações escritas em Javascript/Typescript
- Criado pelo Facebook
- Utilizado para testes de frontend (React, Angular, Vue.js e etc)
- ... e também para backend (NodeJS)



Qualidade de Software

Instalação

```
npm i -D jest
```

Nome dos Arquivos

my-test.spec.js

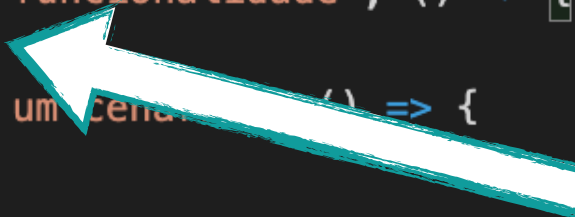
Nome dos Arquivos

my-test.spec.js OU my-test.test.js

<https://jestjs.io/docs/configuration#testmatch-arraystring>

Qualidade de Software

```
describe("Descrição da funcionalidade", () => {  
  test("Descrição de um cenário", () => {  
    // todo  
  });  
  
  test("0, [0] deve gerar exceção", () => {  
    // todo  
  });  
})
```



Descreve a funcionalidade
a ser testada

Qualidade de Software

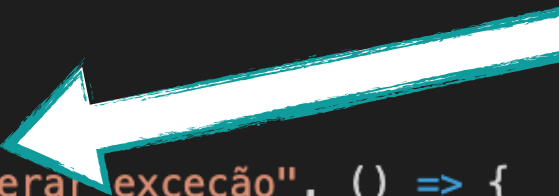
```
describe("Descrição da funcionalidade", () => {  
  
  test("Descrição de um cenário", () => {  
    // todo  
  });  
  
  test("0, [0] deve gerar exceção", () => {  
    // todo  
  });  
})
```



Descreve um cenário
de teste

Qualidade de Software

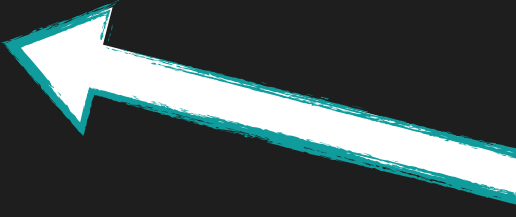
```
describe("Descrição da funcionalidade", () => {  
  
  test("Descrição de um cenário", () => {  
    // todo  
  });  
  
  test("0, [0] deve gerar exceção", () => {  
    // todo  
  });  
})
```



Podemos ter vários cenários de teste para uma funcionalidade

Qualidade de Software

```
describe("Descrição da funcionalidade", () => {  
  beforeEach(() => {  
    // todo  
  })  
  
  beforeEach(() => {  
    // todo  
  })  
  
  test("Descrição de um cenário", () => {  
    // todo  
  });  
})
```



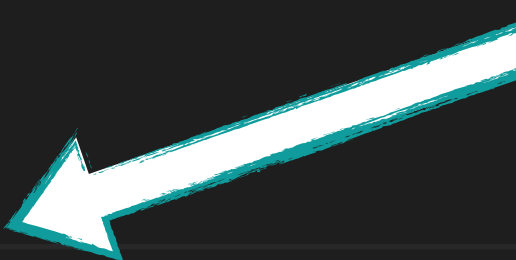
Executa apenas uma vez
antes da execução dos
testes

Utilizado para configurar
todos os cenários

Ex. Abrir conexão com o
banco de dados

Qualidade de Software

```
describe("Descrição da funcionalidade", () => {  
  beforeAll(() => {  
    // todo  
  })  
  
  beforeEach(() => {  
    // todo  
  })  
  
  test("Descrição de um cenário", () => {  
    // todo  
  });  
})
```



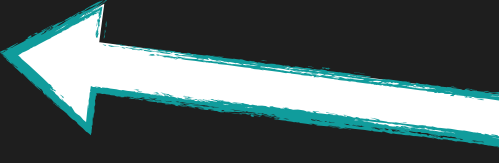
Executa antes de cada teste

Utilizado para configurar cada cenário antes de executa-lo

Ex. limpar a base de dados

Qualidade de Software

```
describe("Descrição da funcionalidade", () => {  
  afterAll(() => {  
    // todo  
  })  
  
  afterEach(() => {  
    // todo  
  })  
  
  test("Descrição de um cenário", () => {  
    // todo  
  });  
})
```



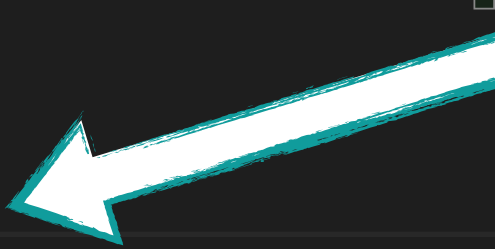
Executa apenas uma vez
após a execução de todos
os testes

Utilizado para limpar
recursos

Ex. Fechar conexão com o
banco de dados

Qualidade de Software

```
describe("Descrição da funcionalidade", () => {  
  afterAll(() => {  
    // todo  
  })  
  
  afterEach(() => {  
    // todo  
  })  
  
  test("Descrição de um cenário", () => {  
    // todo  
  });  
})
```



Executa após a execução de cada teste

Utilizado para limpar recursos entre cada teste

Ex. limpar a base de dados

Asserções

```
expect(resultadoAtual).toBe(resultadoEsperado)
```

```
expect(1 + 1).toBe(2)
```

Qualidade de Software

Asserções

```
describe("Descrição da funcionalidade", () => {  
  test("Descrição de um cenário", () => {  
    expect(1 + 1).toBe(2)  
  });  
})
```


Asserções

```
expect(resultadoAtual).toStrictEqual(resultadoEsperado)
```

```
expect([0, 1]).toStrictEqual([0, 1])
```

Asserções

```
expect(resultadoAtual).not.toBe(resultadoEsperado)
```

```
expect(1 + 1).not.toBe(3)
```

Asserções

```
const expression = () => findUser(-1)  
expect(expression).toThrow('User not Found')  
expect(expression).toThrow('User not Found')
```

Datasets

```
const dataset = [[1,2,3], [2,3,5], [3,5,8]];

test.each(dataset)('%d plus %d should be %d', (a,b,c) => {
  expect(a+b).toBe(c);
});
```

Datasets

```
const dataset = [{a:1,b:2,c:3}, {a:2,b:3,c:5}, {a:3,b:5,c:8}];  
  
test.each(dataset)('$a plus $b should be $c', (a,b,c) => {  
  expect(a+b).toBe(c);  
});
```

Qualidade de Software

Execução

`npx jest`

Introdução ao Jest

PUC Minas **Virtual**

Execução

`npx jest` OU `npm run test`

Qualidade de Software

```
{  
  "name": "pratica-desafio",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  ▶ Debug  
  "scripts": {  
    "test": "jest"  
  },  
  "author": "",  
  "license": "ISC",  
  "devDependencies": {  
    "jest": "^27.3.1"  
  }  
}
```

Execução

npm run test



Qualidade de Software

```
PASS ./two-sum.spec.js
```

```
Two Sum Problema
```

- ✓ 0, [] deve gerar exceção (8 ms)
- ✓ 0, [0] deve gerar exceção (1 ms)
- ✓ 0, [0, 0] deve deve retornar [0, 0] (1 ms)
- ✓ 1, [0, 1] deve retornar [0, 1]
- ✓ 1, [1, 0] deve retornar [1, 0] (1 ms)
- ✓ 5, [0, 1, 2, 3, 4, 5] deve retornar [1, 4]
- ✓ -1, [-2, 1] deve retornar [-2, 1]
- ✓ 5, [1, 3, 8, 10] => deve retornar []

```
Test Suites: 1 passed, 1 total
```

```
Tests: 8 passed, 8 total
```

```
Snapshots: 0 total
```

```
Time: 0.23 s, estimated 1 s
```

```
Ran all test suites.
```

Saída

Qualidade de Software

Execução

```
npm run test -- --watch
```

Cobertura de Código

```
npm run test -- --coverage
```

Qualidade de Software

PASS ./two-sum.spec.js

Two Sum Problema

- ✓ 0, [] deve gerar exceção (9 ms)
- ✓ 0, [0] deve gerar exceção
- ✓ 0, [0, 0] deve deve retornar [0, 0] (1 ms)
- ✓ 1, [0, 1] deve retornar [0, 1]
- ✓ 1, [1, 0] deve retornar [1, 0]
- ✓ 5, [0, 1, 2, 3, 4, 5] deve retornar [1, 4] (1 ms)
- ✓ -1, [-2, 1] deve retornar [-2, 1]
- ✓ 5, [1, 3, 8, 10] => deve retornar []

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	100	100	100	100	
two-sum.js	100	100	100	100	

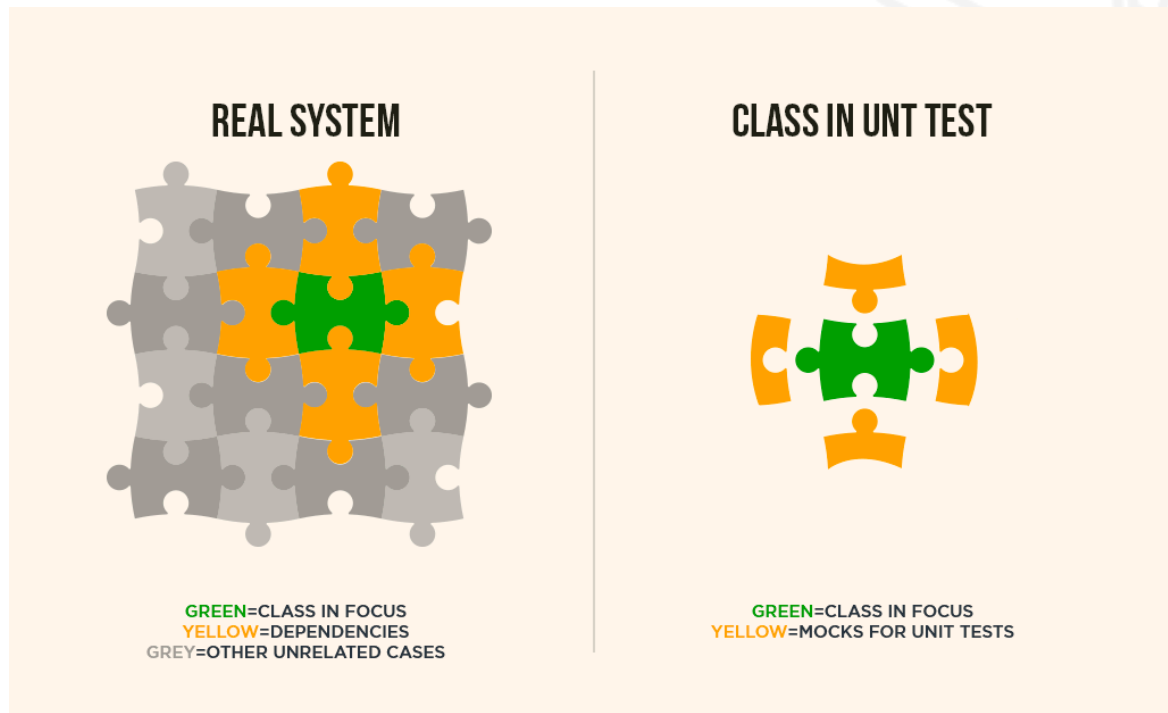
Cobertura de Código

Documentação

jestjs.io

Teste de Unidade com Dublês (Mock)

Qualidade de Software



Qualidade de Software

- Isolamento de recursos externos
- Controle total da unidade dependente
- Simulação do comportamento da dependência

Qualidade de Software

```
describe("UserRepository", () => {  
  let collection;  
  let repository;  
  
  beforeEach(() => {  
    collection = {  
      findOne: jest.fn()  
    }  
    repository = new UserRepository(collection)  
  })  
  
  test("Deve retornar usuário com id 1", () => {  
    collection.findOne.mockReturnValue({  
      id: 1,  
      name: 'John Doe'  
    })  
  
    const user = repository.find(1)  
    expect(user).toStrictEqual({  
      id: 1,  
      name: 'John Doe'  
    })  
  });  
});
```

Declara as variáveis utilizadas em todos os testes

Qualidade de Software

```
describe("UserRepository", () => {  
  
  let collection;  
  let repository;  
  
  beforeEach(() => {  
    collection = {  
      findOne: jest.fn()  
    }  
    repository = new UserRepository(collection)  
  })  
  
  test("Deve retornar usuário com id 1", () => {  
    collection.findOne.mockReturnValue({  
      id: 1,  
      name: 'John Doe'  
    })  
  
    const user = repository.find(1)  
    expect(user).toStrictEqual({  
      id: 1,  
      name: 'John Doe'  
    })  
  });  
});
```

Cria uma nova instancia das variáveis antes de executar cada teste.

Qualidade de Software

```
describe("UserRepository", () => {  
  
  let collection;  
  let repository;  
  
  beforeEach(() => {  
    collection = {  
      findOne: jest.fn()  
    }  
    repository = new UserRepository(collection)  
  })  
  
  test("Deve retornar usuário com id 1", () => {  
    collection.findOne.mockReturnValue({  
      id: 1,  
      name: 'John Doe'  
    })  
  
    const user = repository.find(1)  
    expect(user).toStrictEqual({  
      id: 1,  
      name: 'John Doe'  
    })  
  });  
});
```

Mock do resultado da chamada findOne() do objeto collection.

Temos controle total do que a collection deve retornar.

Qualidade de Software

```
describe("UserRepository", () => {  
  
  let collection;  
  let repository;  
  
  beforeEach(() => {  
    collection = {  
      findOne: jest.fn()  
    }  
    repository = new UserRepository(collection)  
  })  
  
  test("Deve retornar usuário com id 1", () => {  
    collection.findOne.mockReturnValue({  
      id: 1,  
      name: 'John Doe'  
    })  
  
    const user = repository.find(1)  
    expect(user).toEqual({  
      id: 1,  
      name: 'John Doe'  
    })  
  });  
});
```

Faz a chamada para o repositório normalmente e verifica se o resultado está conforme esperado

Espiões (Spy)

Qualidade de Software

- Ferramenta para registrar as interações com os dublês
- Métodos invocados
- Argumentos passados
- Valores retornados

Qualidade de Software

```
test("Deve retornar usuário com id 1", () => {  
  collection.findOne.mockReturnValue({  
    id: 1,  
    name: 'John Doe'  
  })  
  
  const user = repository.find(1)  
  
  expect(user).toStrictEqual({  
    id: 1,  
    name: 'John Doe'  
  })  
  
  expect(collection.findOne).toHaveBeenCalledTimes(1)  
});
```

Verifica se o método
“findOne” foi invocado
fornecendo “1” como
argumento



Teste de Integração



Qualidade de Software

- Testar a comunicação da aplicação com banco de dados real
- Não utilizamos dublês para simular a comunicação
- Possíveis erros de conexão com o banco de dados vão fazer o teste falhar

Qualidade de Software

da teoria à
prática 

Qualidade de Software

Prática 1 - Event Mongo App

Escreva uma classe em Javascript que efetue as operações básicas de banco de dados (ie. CRUD) para um sistema de eventos.

- REQ 1: Criar um novo evento (C - Create)
- REQ 2: Pesquisar evento por nome (R - Read)
- REQ 3: Atualizar um evento (U - Update)
- REQ 4: Remover um evento (D - Delete)

Qualidade de Software

da teoria à
prática 

Qualidade de Software

Prática 2 - User Mongo App

Escreva uma classe em Javascript que efetue as operações básicas de banco de dados (ie. CRUD) para um sistema de usuários.

- REQ 1: Criar um novo usuário (C - Create) - Nome, Email
- REQ 2: Pesquisar usuário por email (R - Read)
- REQ 3: Atualizar um usuário por ID (U - Update)
- REQ 4: Remover um usuário por ID (D - Delete)



PUC Minas
Virtual