

Fundamentos de teste de software

Júlia Luiza

jlslc@icomp.ufam.edu.br

Cronograma: Visão Geral

- **Visão geral:**
 - Porque é importante testar? Quando testar?;
 - Diferentes tipos de testes;
 - Ferramentas disponíveis.
- **Framework Jest**
 - Introdução (o que é, vantagens, etc.);
 - Instalação do Jest usando npm;
 - Configuração do ambiente de teste (arquivo `jest.config.js`, script para executar testes).
- **Testes unitários no back-end com Jest**
 - Como escrever testes (o que testar, sintaxe e exemplos);
 - Prática com testes unitários para back-end em JS;
 - Cobertura de código e métricas;
 - Boas práticas em testes unitários;

25/07 (terça)

Cronograma: Visão Geral

- Recapitação da aula anterior
- **Testes de integração no back-end com Jest e Supertest**
 - Como escrever testes de integração (o que testar, sintaxe, exemplos e atividades);
 - Prática com testes automatizados para requisições (camada serviço) do projeto em andamento, utilizando Supertest e banco de dados de teste;
 - Boas práticas em testes de integração;
- **Testes com Jest no front-end com React**
 - Instalação do Jest em um projeto React;
 - Como escrever testes unitários utilizando Jest e react-testing-library (o que testar, sintaxe, ferramentas e exemplos);
 - Prática com testes unitários para o front-end do projeto em andamento;
 - Boas práticas de testes unitários no front-end.

27/07 (quinta)

Avaliações

Questionários no colabweb

- Aula 1 -> Questionário 1
- Aula 2 -> Questionário 2
- Média questionários = N1

Trabalho prático

- Testes unitários/integração do projeto em andamento lojaVirtual
- Prazo: a definir
- Nota trabalho = N2

$$\text{Média disciplina} = (N1 + N2) / 2$$

Cronograma: Aula 01

Visão geral

- Por que é importante testar?
- Diferentes tipos de teste;
- Ferramentas disponíveis.

Jest

- O que é e vantagens;
- Instalação utilizando npm;
- Configurações adicionais.

Testes no back-end

- Como escrever testes;
- Prática com testes para o back-end do projeto em andamento;
- Cobertura de código;
- Boas práticas.

Motivação para testar



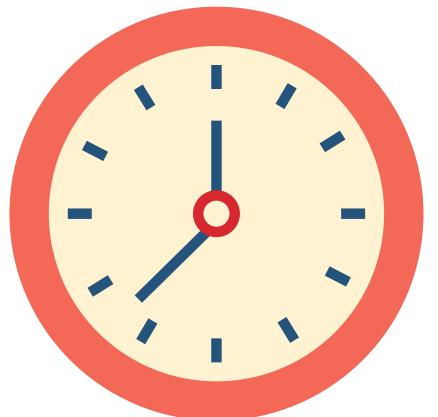
Entusiasmo

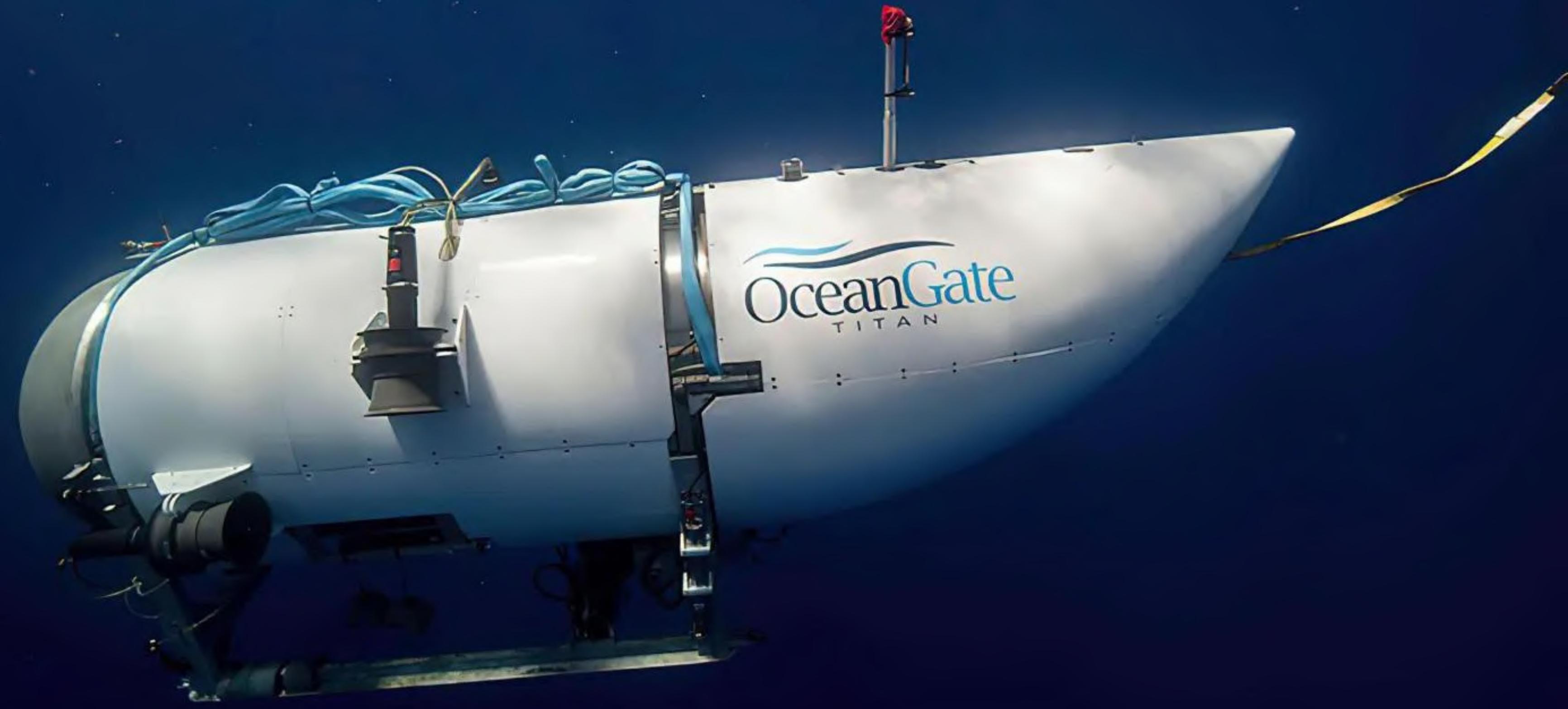
Por que testar software?

- Qualidade;
- Consistência;
- Manutenibilidade;
- Confiabilidade;
- Evolução do software;
- Evitar que erros cheguem até o usuário final.

Por que testar software?

- Qualidade;
- Consistência;
- Manutenibilidade;
- Confiabilidade;
- Evolução do software;
- Evitar que erros cheguem até o usuário final.





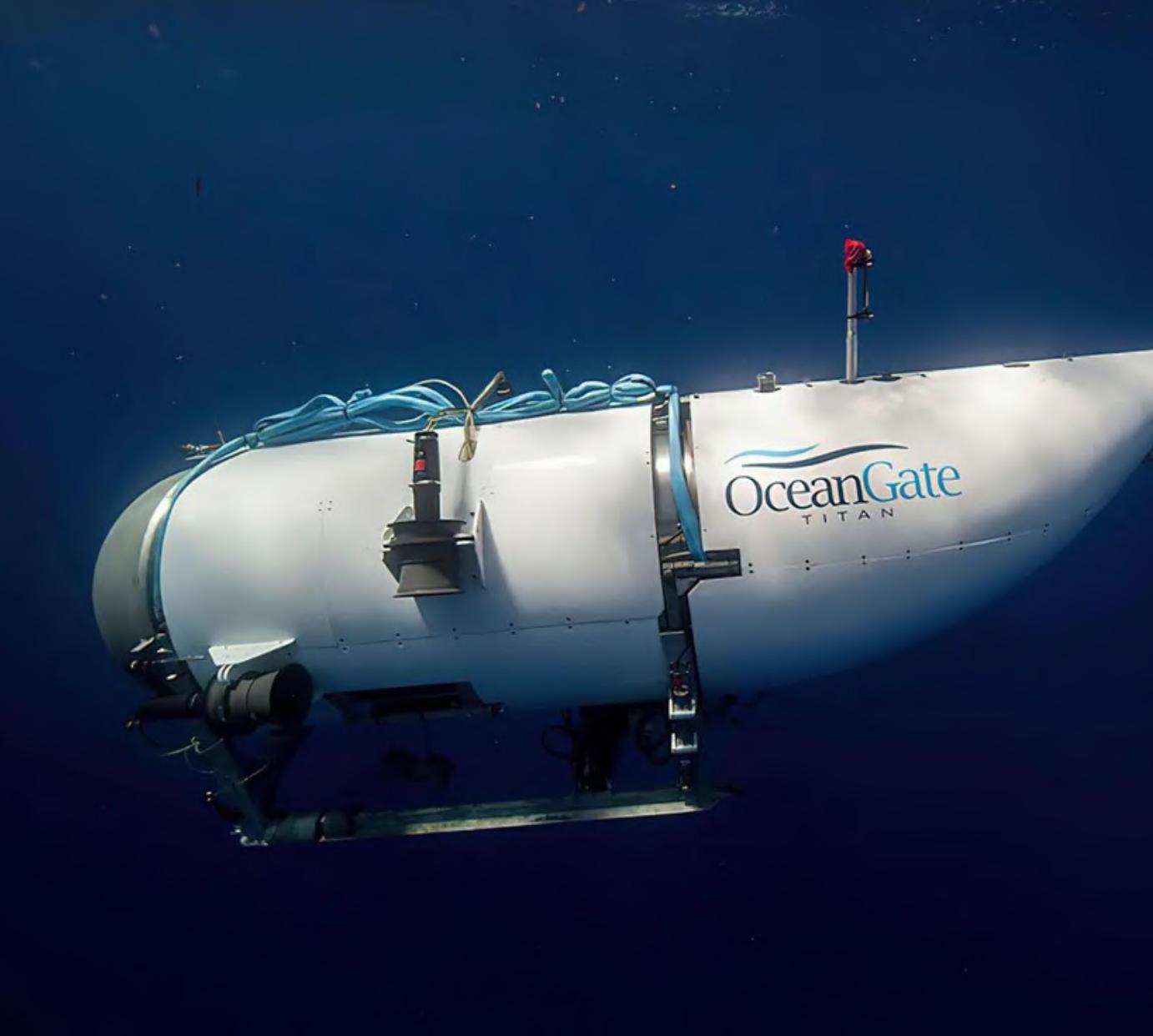
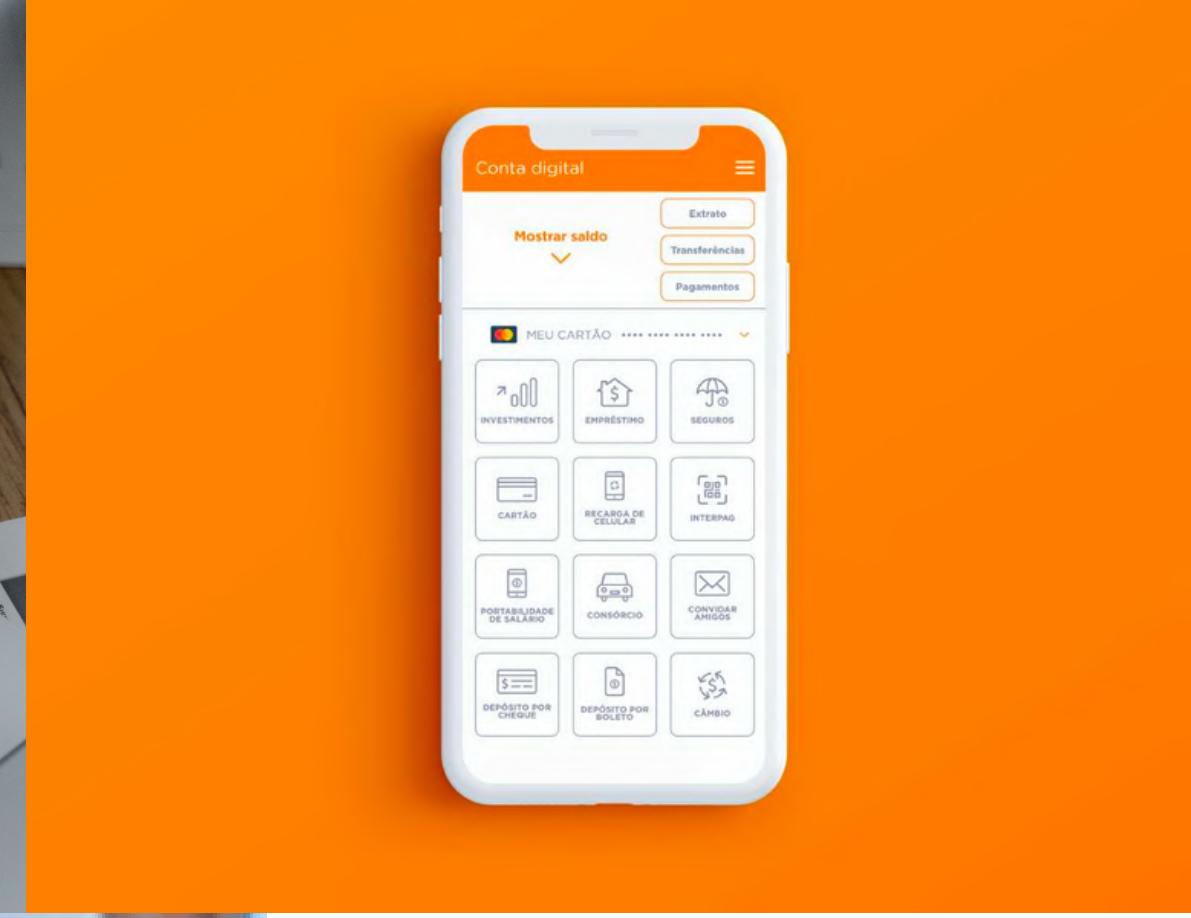
Quando testar?



ATENÇÃO

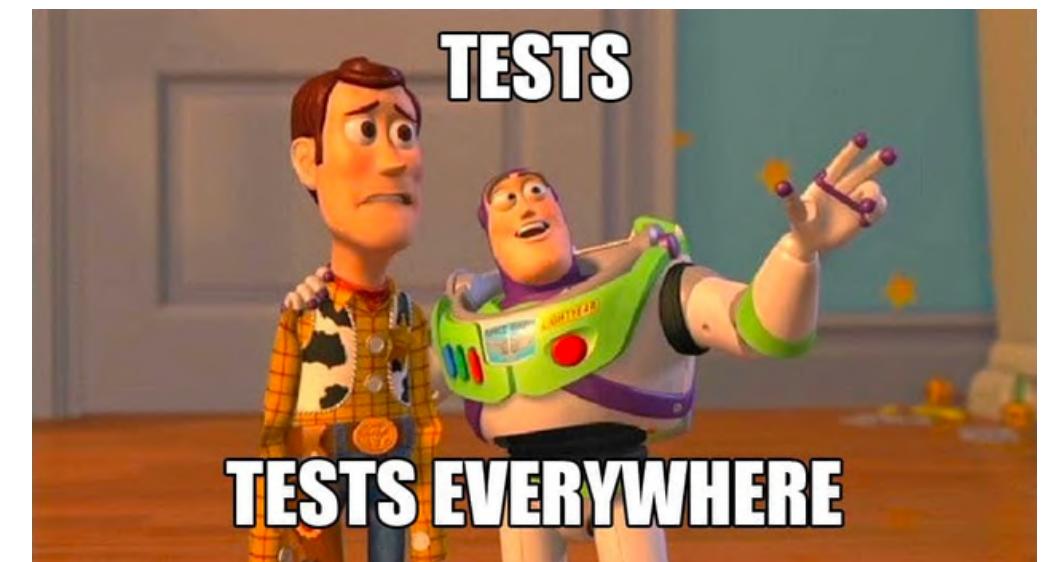






Quando testar?

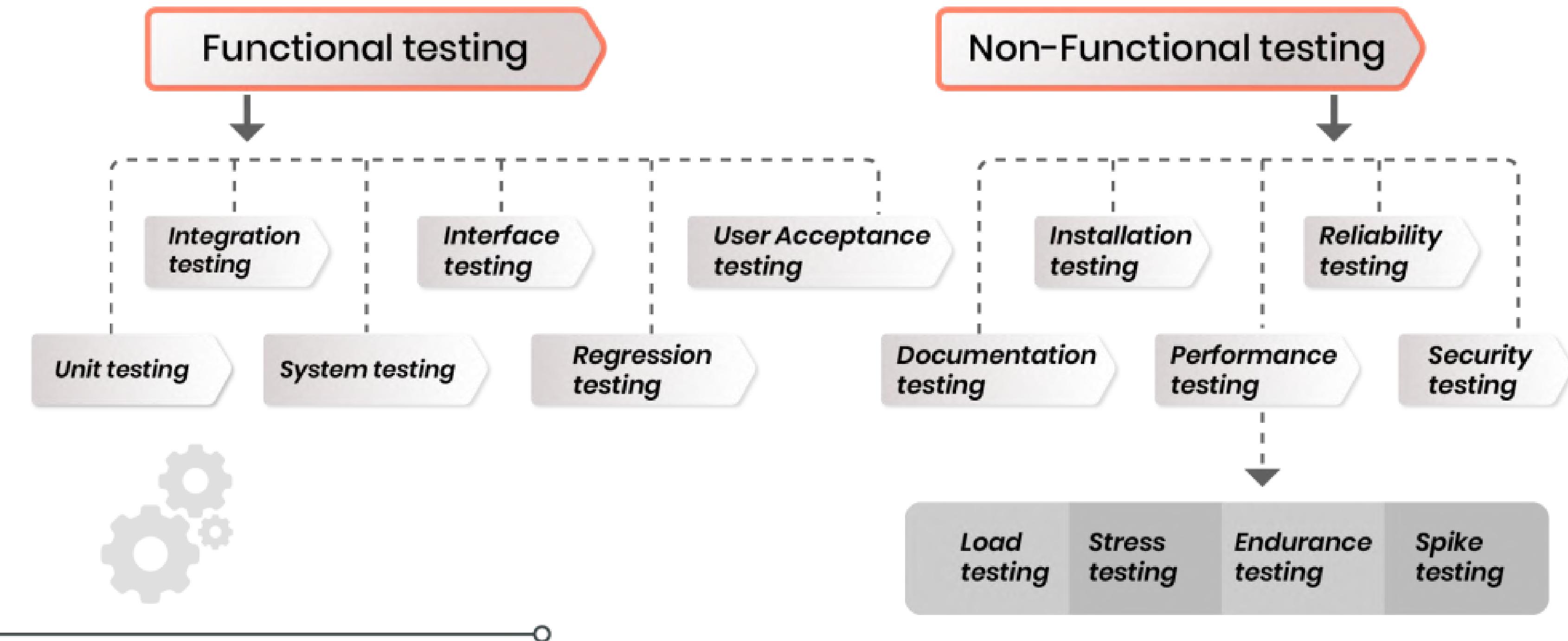
- **Sempre que possível:**
 - Antes de desenvolver;
 - Durante o desenvolvimento;
 - Depois do lançamento.
- Testes unitários e de integração:
 - Durante o desenvolvimento de novas features;
 - Durante a manutenção de features existentes;



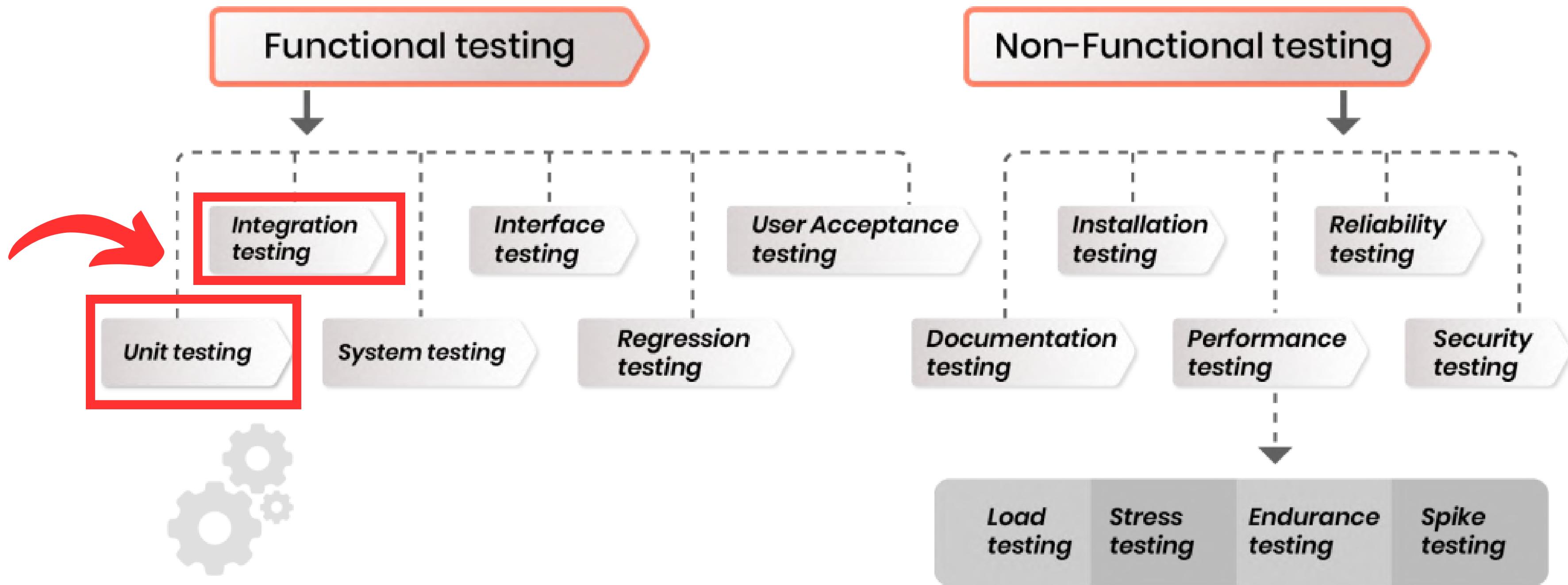
Diferentes tipos de teste



TYPES OF SOFTWARE TESTING



TYPES OF SOFTWARE TESTING





Comprar novamente

[Ver todos e gerenciar](#)

Umidificador de Ar Elgin Digital Inteligente, 2,5L
R\$ 189,99

✓prime

[Adicionar ao carrinho](#)

Cola Puzzle Brilhante
R\$ 28,99

✓prime

[Adicionar ao carrinho](#)

Antipulgas Zoetis Simparic 40 Mg Para Cães
R\$ 219,90
(R\$ 73,30/unidade)

✓prime

[Adicionar ao carrinho](#)

Suas listas

[Lista de compras](#)[Cha de bebe da Erica](#)[Márcia](#)[Criar uma Lista de desejos](#)

Sua conta

[Sua conta](#)[Seus pedidos](#)[Sua Lista de desejos](#)[Recomendados para você](#)[Programe e Poupe](#)[Sua assinatura Prime](#)[Inscrições e assinaturas](#)[Gerencie seu conteúdo e dispositivos](#)[Seu Amazon Music](#)[Seu Prime Video](#)[Seu Kindle Unlimited](#)[Seu Amazon Drive em Amazon.com](#)[Seus aplicativos e dispositivos](#)[Trocarn de conta](#)[Sair da conta](#)

Continue de onde parou

[Apple iPhone SE \(3ª geração\) - Apple iPhone 14 \(128GB\)](#)[Apple iPhone 14 Pro \(128GB\) - Kit Capa Anti Impacto](#)

Continuar comprando

[Ebooks kindle
Visto 2x](#)[Malas de bordo
Visto 1x](#)[Kits de brinquedos diferentes
Visto 5x](#)[Colchonetes para ex...
Visto 2x](#)[Visualize seu histórico de navegação](#)

Compre novamente

[Ver todos e gerenciar](#)

Umidificador de Ar Elgin Digital Inteligente, 2,5L
R\$ 189,99

✓prime

[Adicionar ao carrinho](#)

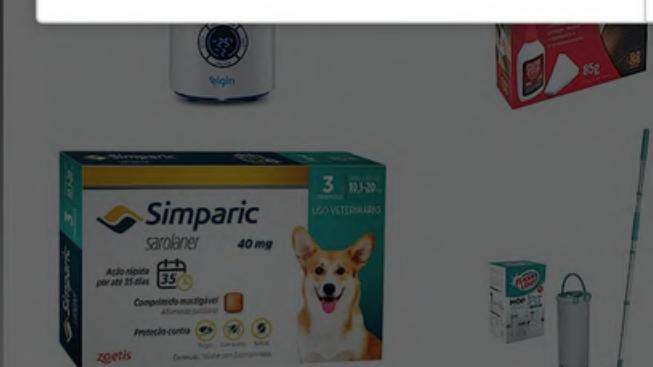
Cola Puzzle Brilhante
R\$ 28,99

✓prime

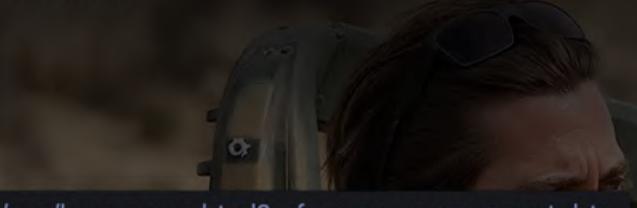
[Adicionar ao carrinho](#)

Antipulgas Zoetis Simparic 40 Mg Para Cães
R\$ 219,90
(R\$ 73,30/unidade)

✓prime

[Adicionar ao carrinho](#)[Compre seus itens essenciais do dia a dia](#)[Ver mais itens para comprar novamente](#)

Prime Video: Recomendado para você

[O Pacto](#)

Ganhe créditos completando missões



Ofertas do Dia





Comprar novamente

Ver todos e gerenciar

Umidificador de Ar Elgin
Digital Inteligente, 2,5L
R\$ 189,99

prime

Adicionar ao carrinho

Cola Puzzle Brilhante
R\$ 28,99

prime

Adicionar ao carrinho

Antipulgas Zoetis
Simparic 40 Mg Para Cães
R\$ 219,90
(R\$ 73,30/unidade)

prime

Adicionar ao carrinho

Suas listas

Lista de compras

Cha de bebe da Erica

Márcia

Criar uma Lista de desejos

Sua conta

Sua conta

Seus pedidos

Sua Lista de desejos

Recomendados para você

Programa e Poupe

Sua assinatura Prime

Inscrições e assinaturas

Gerencie seu conteúdo e dispositivos

Seu Amazon Music

Seu Prime Video

Seu Kindle Unlimited

Seu Amazon Drive em Amazon.com

Seus aplicativos e dispositivos

Trocando de conta

Sair da conta



Continue de onde parou



Apple iPhone SE (3ª geração) Apple iPhone 14 (128GB)



Apple iPhone 14 Pro (128GB) Kit Capa Anti Impacto

Continuar comprando

Ebooks kindle
Visto 2xMalas de bordo
Visto 1xKits de brinquedos diferentes
Visto 5xColchonetes para exteriores
Visto 2x

Visualize seu histórico de navegação

Comprar novamente

Ver todos e gerenciar

Umidificador de Ar Elgin
Digital Inteligente, 2,5L
R\$ 189,99

prime

Adicionar ao carrinho

Cola Puzzle Brilhante
R\$ 28,99

prime

Adicionar ao carrinho

Antipulgas Zoetis
Simparic 40 Mg Para Cães
R\$ 219,90
(R\$ 73,30/unidade)

prime

Adicionar ao carrinho



Compre seus itens essenciais do dia a dia

Ver mais itens para comprar novamente

Paramount+
ORIGINAL
ASSINE JÁ
prime video | CHANNELS
VERIFIQUE A CLASSIFICAÇÃO INDICATIVA

Patrocinado

Prime Video: Recomendado para você

O Pacto



Ganhe créditos completando missões



Ofertas do Dia





Vendas na Amazon Ofertas do Dia eBooks Kindle Ferramentas e Construção Atendimento ao Cliente Comprar novo



Continue de onde parou



Apple iPhone SE (3^a ger... Apple iPhone 14 (128...



Apple iPhone 14 Pro (1... Kit Capa Anti Impacto...

Continuar comprando



Ebooks kindle Malas de bordo

Comprar novamente

[Ver todos e gerenciar](#)



Umidificador de Ar Elgin Digital Inteligente, 2,5...
R\$ 189,99

✓prime

[Adicionar ao carrinho](#)



Cola Puzzle Brilhante
R\$ 28,99

✓prime

[Adicionar ao carrinho](#)



Antipulgas Zoetis Simparic 40 Mg Para C...
R\$ 219,90
(R\$ 73,30/unidade)

✓prime

[Adicionar ao carrinho](#)

Suas listas

[Lista de compras](#)

Cha de bebe da Erica

Márcia

[Criar uma Lista de desejos](#)

Sua conta

[Sua conta](#)

Seus pedidos

Sua Lista de desejos

Recomendados para você

Programe e Poupe

Sua assinatura Prime

Inscrições e assinaturas

Gerencie seu conteúdo e dispositivos

Seu Amazon Music

Seu Prime Video

Seu Kindle Unlimited

Seu Amazon Drive em Amazon.com

Seus aplicativos e dispositivos

Trocar de conta

Sair da conta

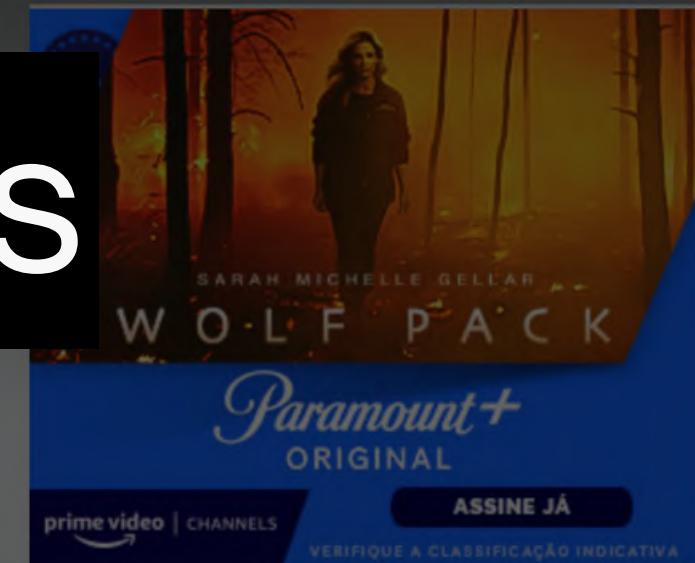
Features isoladas

Compre seus itens essenciais do dia a dia

Kits de brinquedos d... Colchonetes para ex...
Visto 5x Visto 2x

[Visualize seu histórico de navegação](#)

[Ver mais itens para comprar novamente](#)



Ofertas do Dia



Prime Video: Recomendado para você

O Pacto

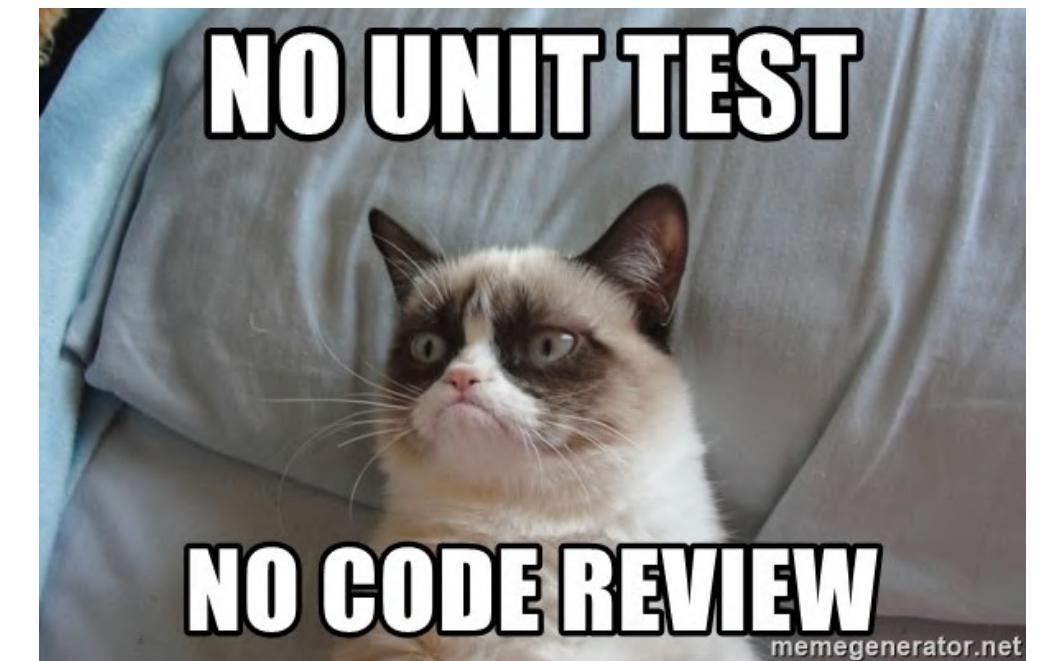


Ganhe créditos completando missões



Testes unitários

- Componentes individuais, com lógicas isoladas e preferencialmente sem *side-effect* são testados;
- **Valida o funcionamento de cada unidade do software, individualmente;**
- Implementados durante o desenvolvimento do software, pelos desenvolvedores;
- Exemplos: funções, métodos, classes, etc.





eBooks Kindle

Ferramentas e Construção

Atendimento ao Cliente

Comprar nov...



Continue de onde parou

Apple iPhone SE (3^a ger...) Apple iPhone 14 (128...

Apple iPhone 14 Pro (1... Kit Capa Anti Impacto...

Continuar comprando



Ebooks kindle Malas de bordo

Kits de brinquedos d... Colchonetes para ex...

Visto 5x Visto 2x

Visualize seu histórico de navegação

Comprar novamente

Ver todos e gerenciar

Umidificador de Ar Elgin
Digital Inteligente, 2,5...
R\$ 189,99

prime

Adicionar ao carrinho

Cola Puzzle Brilhante
R\$ 28,99

prime

Adicionar ao carrinho

Antipulgas Zoetis
Simparic 40 Mg Para C...
R\$ 219,90
(R\$ 73,30/unidade)

prime

Adicionar ao carrinho

Suas listas

Lista de compras

Cha de bebe da Erica

Márcia

Criar uma Lista de desejos

Sua conta

Sua conta

Seus pedidos

Sua Lista de desejos

Recomendados para você

Programa e Poupe

Sua assinatura Prime

Inscrições e assinaturas

Gerencie seu conteúdo e dispositivos

Seu Amazon Music

Seu Prime Video

Seu Kindle Unlimited

Seu Amazon Drive em Amazon.com

Seus aplicativos e dispositivos

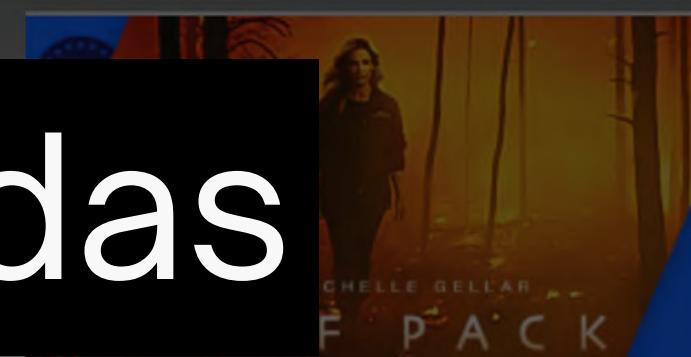
Trocá de conta

Sair da conta

Features integradas

Compre seus itens essenciais do dia a dia

Ver mais itens para comprar novamente

Paramount+
ORIGINAL
ASSINE JÁ
prime video | CHANNELS
VERIFIQUE A CLASSIFICAÇÃO INDICATIVA

Patrocinado

Prime Video: Recomendado para você

O Pacto



Ganhe créditos completando missões



Ofertas do Dia

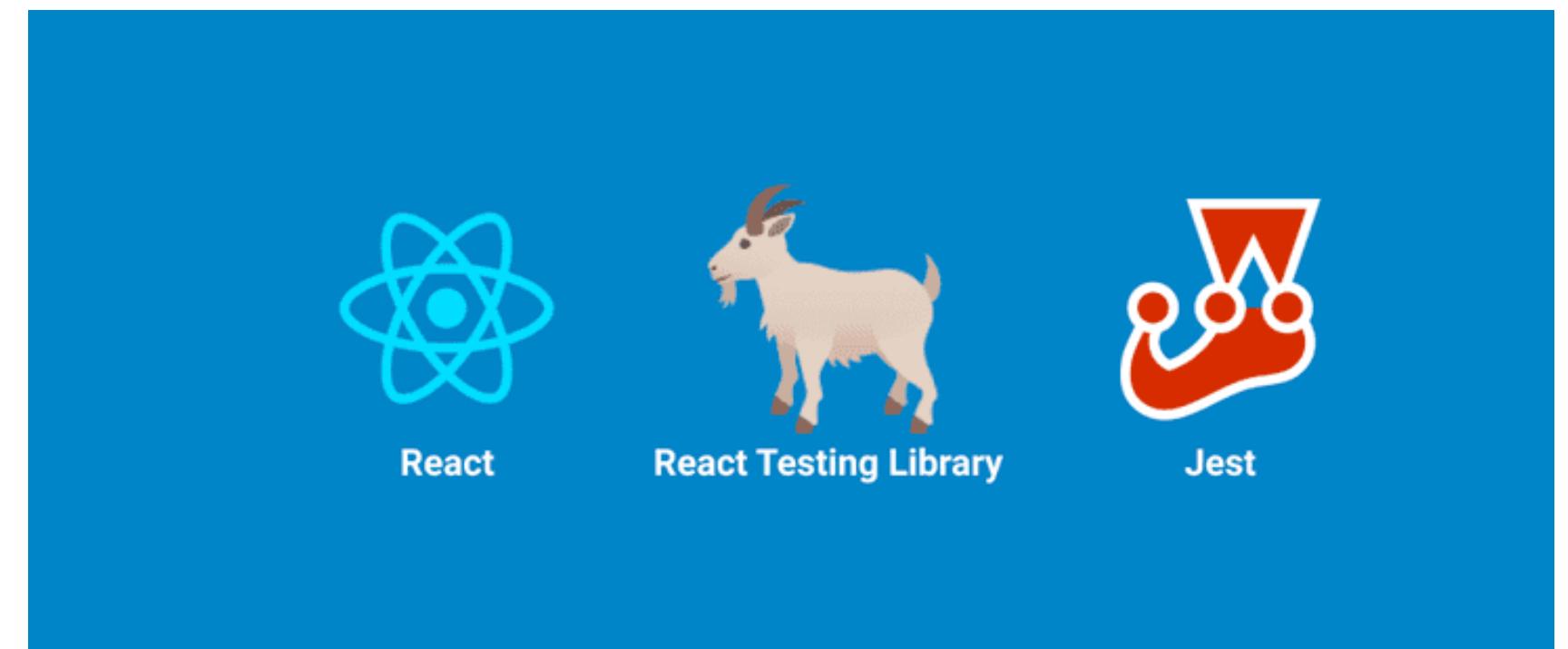
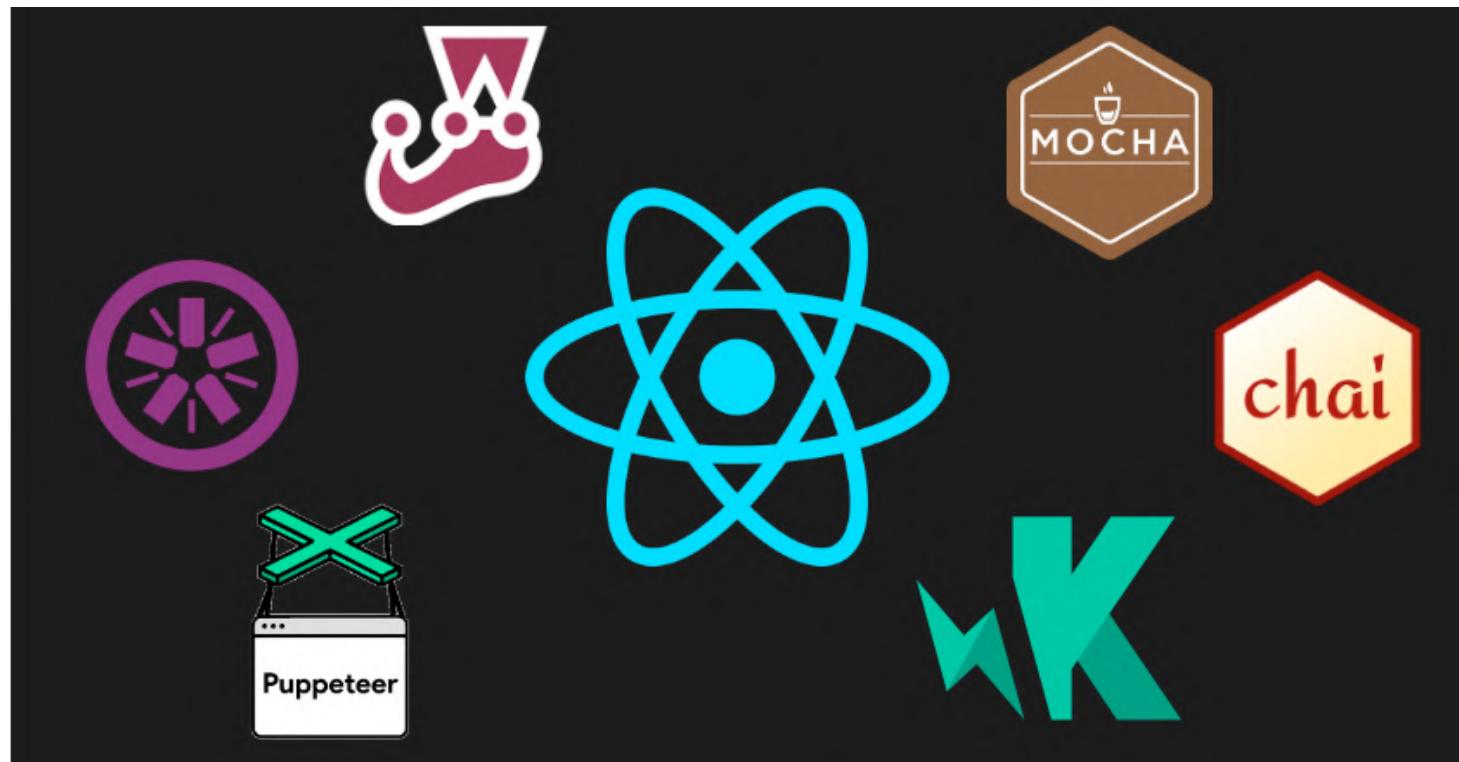
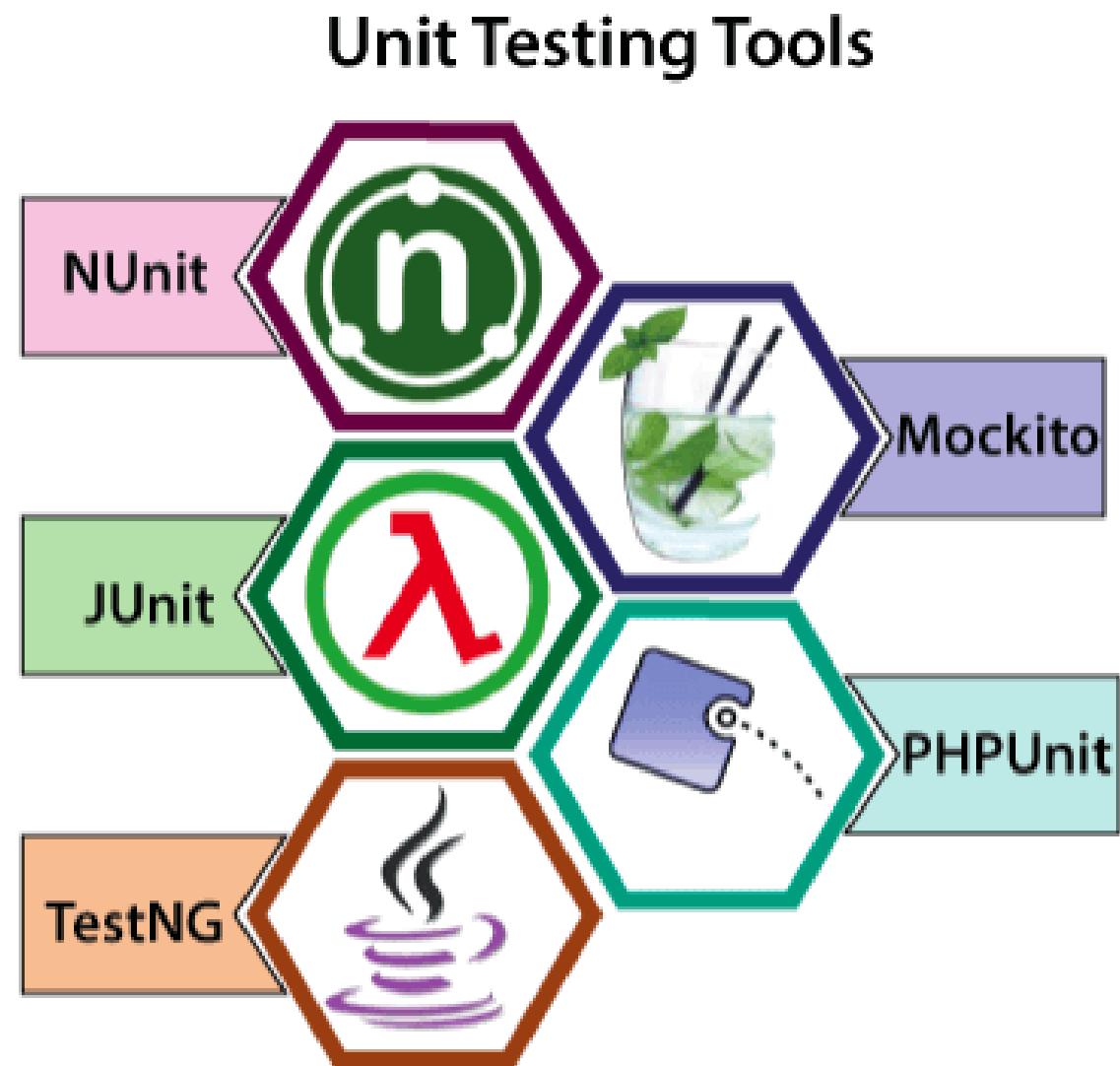


Testes de integração

- Múltiplos componentes, com lógicas integradas e *side-effect* são testados;
- **Valida o funcionamento das unidades de software de forma integrada;**
- Complementares aos testes unitários;
- Exemplos: chamada de um módulo para outro, chamada externa, acesso ao banco de dados, acesso a API.



Ferramentas



Cronograma: Aula 01



ATENÇÃO

Dúvidas?

Visão geral

- Por que é importante testar?
- Diferentes tipos de teste;
- Ferramentas disponíveis.



Jest

- O que é e vantagens;
- Instalação utilizando npm;
- Configurações adicionais.

Testes no back-end

- Como escrever testes;
- Prática com testes para o back-end do projeto em andamento;
- Cobertura de código;
- Boas práticas.

"Jest é um poderoso Framework
de Testes em JavaScript com um
foco na simplicidade."

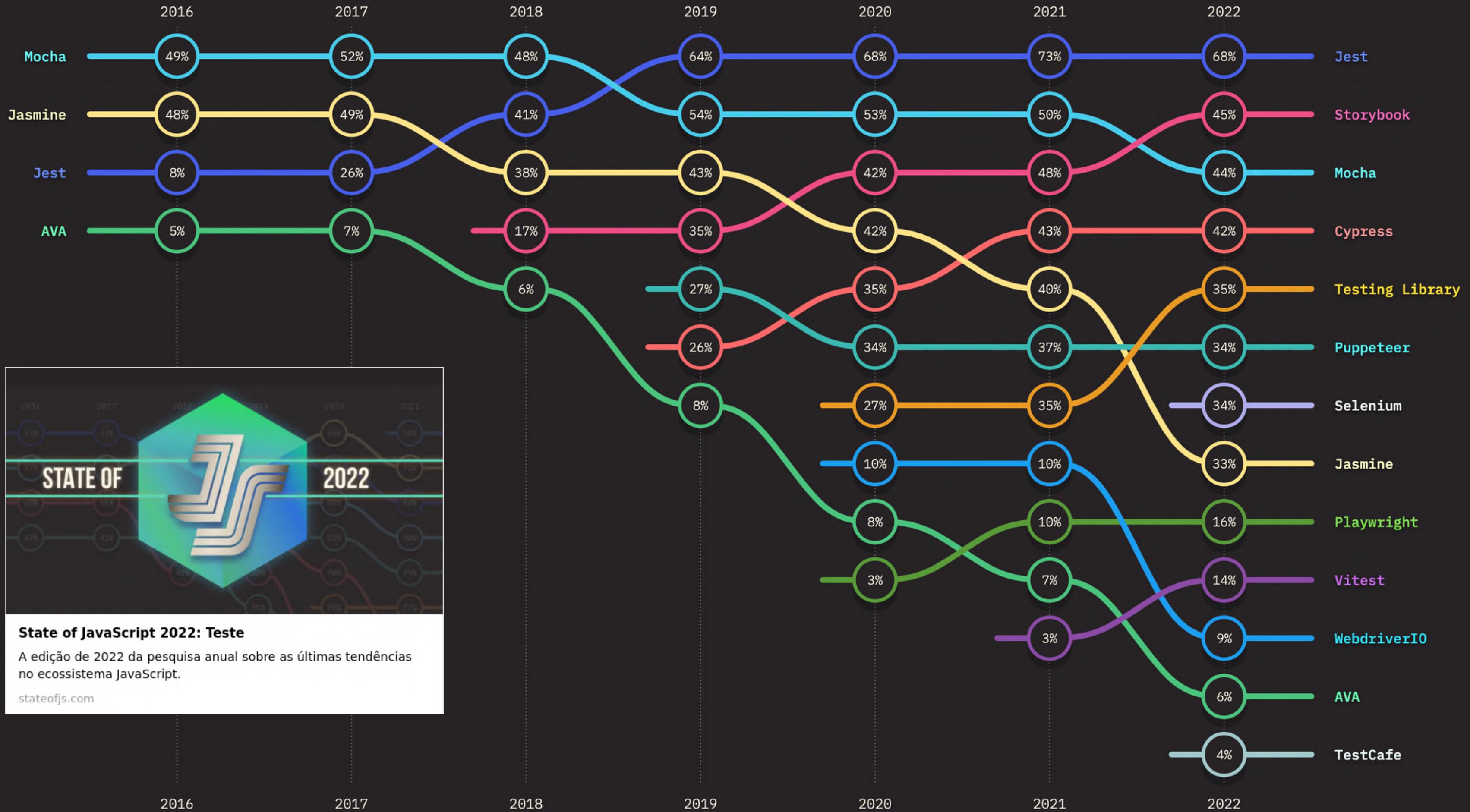
— <https://jestjs.io/pt-BR/>



State of JavaScript 2022: Teste

A edição de 2022 da pesquisa anual sobre as últimas tendências no ecossistema JavaScript.

stateofjs.com



Jest

zero configuração

Jest visa trabalhar fora da caixa, sem configuração, na maioria dos projetos JavaScript.

isolado

Os testes são paralelos e executados em seus próprios processos para maximizar o desempenho.

excelente api

De `it` para `expect` - Jest tem todo o conjunto de ferramentas em um só lugar. Bem documentado e mantido.

Install

```
> npm i jest
```

Repository

❖ github.com/facebook/jest

Homepage

⌚ jestjs.io/

Weekly Downloads

21.722.310



<https://www.npmjs.com/package/jest?activeTab=readme>

RÁPIDO E SEGURO

```
PASS packages/diff-sequences/src/_tests__/index.test.js
PASS packages/jest-diff/src/_tests__/diff.test.js
PASS packages/jest-mock/src/_tests__/jest_mock.test.js
PASS packages/jest-util/src/_tests__/fakeTimers.test.js
PASS packages/pretty-format/src/_tests__/prettyFormat.test.js

RUNS packages/jest-haste-map/src/_tests__/index.test.js
RUNS packages/pretty-format/src/_tests__/DOMElement.test.js
RUNS packages/jest-config/src/_tests__/normalize.test.js
RUNS packages/expect/src/_tests__/matchers.test.js
RUNS packages/pretty-format/src/_tests__/Immutable.test.js
RUNS packages/expect/src/_tests__/spyMatchers.test.js
RUNS packages/jest-cli/src/_tests__/SearchSource.test.js
RUNS packages/jest-runtime/src/_tests__/script_transformer.test.js
RUNS packages/jest-cli/src/_tests__/watch.test.js
RUNS packages/jest-haste-map/src/crawlers/_tests__/watchman.test.js
RUNS packages/pretty-format/src/_tests__/react.test.js

Test Suites: 5 passed, 5 of 303 total
Tests:      332 passed, 332 total
```

Ao garantir que seus testes têm um estado global único, Jest pode executar testes em paralelo de forma confiável. Para tornar as coisas rápidas, Jest executa testes anteriores falharam primeiro e organiza novamente com base no quanto os arquivos de teste demoram.

COBERTURA DE CÓDIGO

Crie relatórios de cobertura de código facilmente usando [--coverage](#). Sem necessidade adicional de configurações ou bibliotecas! Jest consegue coletar informações de cobertura de código de projetos inteiros, incluindo arquivos não testados.

```
~/d/p/j/e/timer $ yarn jest --coverage
yarn run v1.12.3
$ /Users/ortatherox/dev/projects/jest/jest/examples/timer/node_modules/.bin/jest --coverage
PASS __tests__/timer_game.test.js
PASS __tests__/infinite_timer_game.test.js
-----|-----|-----|-----|-----|-----|
File    | %Stmts | %Branch | %Funcs | %Lines | Uncovered Line #s |
-----|-----|-----|-----|-----|-----|
All files | 87.5 | 100 | 80 | 87.5 |
infiniteTimerGame.js | 80 | 100 | 66.67 | 80 | 12 |
timerGame.js | 100 | 100 | 100 | 100 |
-----|-----|-----|-----|-----|-----|
Test Suites: 2 passed, 2 total
Tests:      4 passed, 4 total
Snapshots:  0 total
Time:       0.878s, estimated 1s
Ran all test suites.
```

```
• toHaveBeenCalledWith two arguments
expect(jest.fn()).toHaveBeenCalledWith(...expected)
Expected: "last", 10
Received:
 1: "first", 1
 2: "second", 2
 3: "third", 3
Number of calls: 9
```

SIMULAÇÕES FÁCEIS

Jest uses a custom resolver for imports in your tests, making it simple to mock any object outside of your test's scope. You can use mocked imports with the rich [Mock Functions API](#) to spy on function calls with readable test syntax.

Jest: como instalar e configurar

```
npm install --save-dev jest
```

- Se for um projeto React iniciado com *create-react-app*, o **Jest** já estará presente por padrão;
- Se for um projeto com typescript, será necessário também instalar o *babel-jest* e outras libs auxiliares:
 - *npm install --save-dev babel-jest @babel/core @babel/preset-env*
 - *npm install --save-dev @babel/preset-typescript*
 - e configurar o arquivo babel.config.js

<https://jestjs.io/pt-BR/docs/getting-started#usando-typescript>

<https://jestjs.io/docs/testing-frameworks>

Jest: como instalar e configurar

```
PS C:\Users\julia\Documents\web_academy_uf\test\LojaVirtualWA\backend> npx jest --init

The following questions will help Jest to create a suitable configuration
for your project

✓ Would you like to use Jest when running "test" script in "package.json"?
  ... yes
✓ Would you like to use Typescript for the configuration file? ... yes
✓ Choose the test environment that will be used for testing » node
✓ Do you want Jest to add coverage reports? ... yes
✓ Which provider should be used to instrument code for coverage? » v8
✓ Automatically clear mock calls, instances, contexts and results before e
very test? ... yes

📝 Modified C:\Users\julia\Documents\web_academy_uf\test\LojaVirtualWA\ba
ckend\package.json

📝 Configuration file created at C:\Users\julia\Documents\web_academy_uf\
test\LojaVirtualWA\backend\jest.config.ts
PS C:\Users\julia\Documents\web_academy_uf\test\LojaVirtualWA\backend>
```

Jest: como instalar e configurar

```
"scripts": {  
  "start": "nodemon -e js,json,ts,yaml src/index.ts",  
  "start:prod": "node build/index.js",  
  "build": "npx tsc",  
  "tsc:status": "tsc --diagnostics",  
  "test": "jest"  
},
```

Jest: como instalar e configurar

Vamos praticar?



1. Clonar repositório https://github.com/julialuiza/myapp_test
2. Instalar Jest utilizando npm
3. Inserir script para rodar testes no package.json
4. Conferir se script está rodando corretamente

Cronograma: Aula 01



ATENÇÃO

Dúvidas?

Visão geral

- Por que é importante testar?
- Diferentes tipos de teste;
- Ferramentas disponíveis.



Jest

- O que é e vantagens;
- Instalação utilizando npm;
- Configurações adicionais.



Testes no back-end

- Como escrever testes;
- Prática com testes para o back-end do projeto em andamento;
- Cobertura de código;
- Boas práticas.

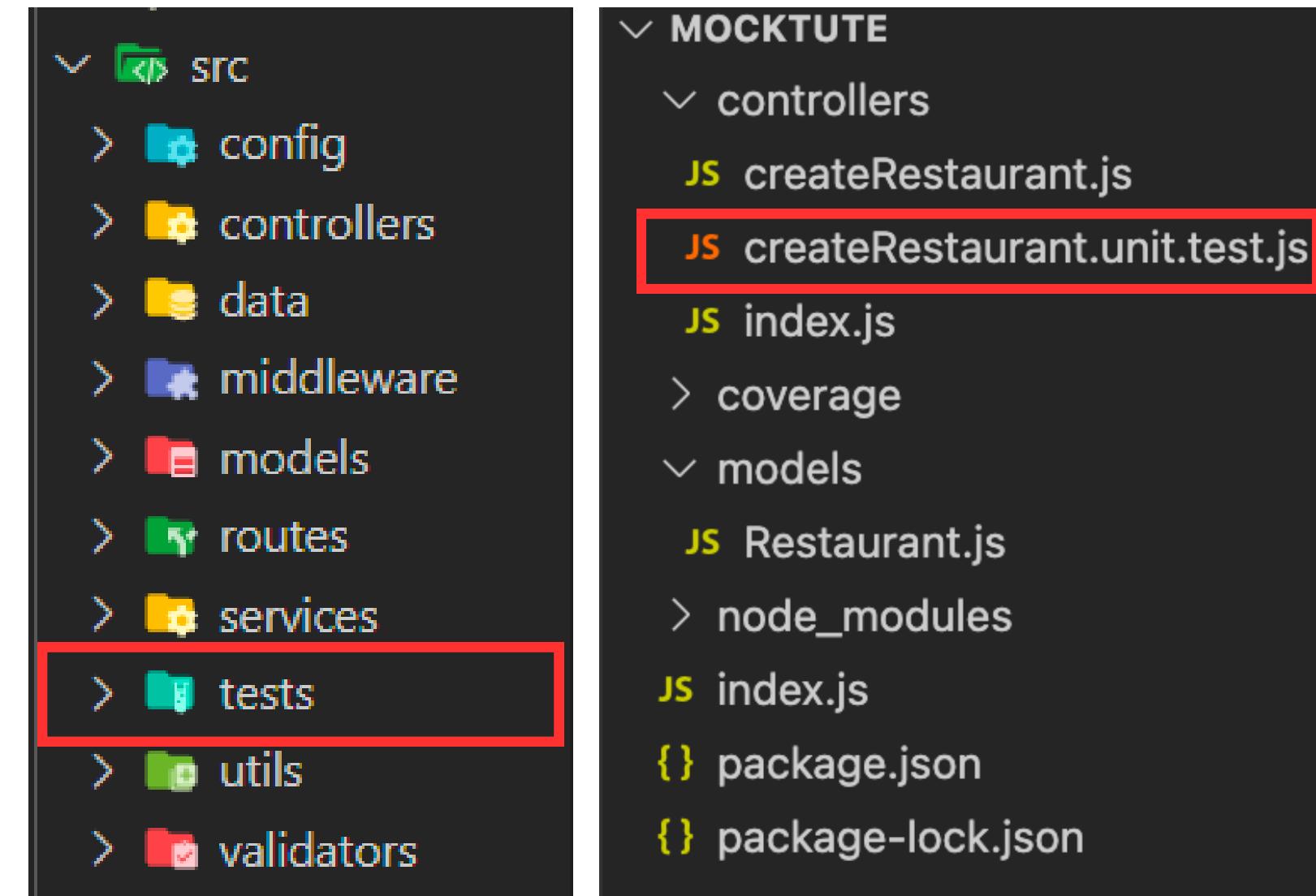
Como escrever testes

Organização:

- *Usualmente*, os arquivos de teste localizam-se perto dos módulos que estão sendo testados;
- Ou em uma pasta dedicada para os testes do projeto;
- No geral, segue-se padrão definido pelo time, empresa, etc.

Arquivo de teste:

- *Usualmente*, cada arquivo de teste comporta testes para um contexto específico do projeto;
- Por padrão, costuma-se nomear com o mesmo nome do arquivo ou função que será testado;
- Ex: itemCarrinho.js -> itemCarrinho.test.js



Como escrever testes

Estrutura:

- O Jest dispõe de diversos recursos para estruturação dos testes, então **a estrutura final irá depender do cenário a ser testado;**
- De modo geral, um teste possui:
 - Um ou mais blocos "**describe**" (descrição do teste)
 - Que comportam um ou mais blocos "**it**" (o teste em si)
- Além disso, existem os blocos:
 - **beforeEach()** -> executado sempre antes de cada bloco **it**
 - **afterEach()** -> executado sempre depois de cada bloco **it**
 - **beforeAll()** -> executado uma vez antes de todos os blocos **it** existentes executarem
 - **afterAll()** -> executado uma vez depois de todos os blocos **it** existentes executarem
 - Obs.: podem ser usados no topo do arquivo de teste ou dentro de cada bloco **describe**

Como escrever testes

```
beforeEach(() => {
  initializeCityDatabase();
});

afterEach(() => {
  clearCityDatabase();
});

test('city database has Vienna', () => {
  expect(isCity('Vienna')).toBeTruthy();
});

test('city database has San Juan', () => {
  expect(isCity('San Juan')).toBeTruthy();
});
```

```
beforeAll(() => {
  return initializeCityDatabase();
});

afterAll(() => {
  return clearCityDatabase();
});

test('city database has Vienna', () => {
  expect(isCity('Vienna')).toBeTruthy();
});

test('city database has San Juan', () => {
  expect(isCity('San Juan')).toBeTruthy();
});
```

Como escrever testes

Sintaxe:

- Agora, para realizar o teste em si, utilizamos o "expect", "modifiers" e "matchers" que o Jest provê:
 - São muitos, então o ideal é estar sempre com a documentação ao lado durante o desenvolvimento dos testes;
- Às vezes, mais de um *matcher* irá servir para a conferência do teste, mas vale conferir qual das opções é a mais semântica para o teste em questão:
 - Exemplo: *toBe(true)* e *toBeTruthy()* podem produzir o mesmo resultado, dependendo do retorno da função testada, mas possuem semântica diferente.

```
drinkSomeLaCroix();
if (thirstInfo()) {
  drinkMoreLaCroix();
}
```

```
test('drinking La Croix leads to having thirst info', () => {
  drinkSomeLaCroix();
  expect(thirstInfo()).toBeTruthy();
});
```

Jest Matchers

```
test('dois mais dois é quatro', () => {
  expect(2 + 2).toBe(4);
});
```

```
test('atribuição de objeto', () => {
  const data = {one: 1};
  data['two'] = 2;
  expect(data).toEqual({one: 1, two: 2});
});
```

```
test('dois mais dois', () => {
  const value = 2 + 2;
  expect(value).toBeGreaterThan(3);
  expect(value).toBeGreaterThanOrEqual(3.5);
};

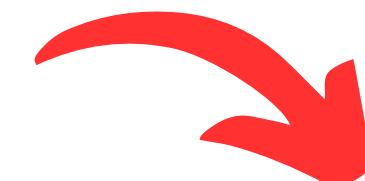
test('não existe I em team', () => {
  expect('team').not.toMatch(/I/);
});
```

```
test('resolves to lemon', async () => {
  await expect(Promise.resolve('lemon')).resolves.toBe('lemon');
  await expect(Promise.resolve('lemon')).resolves.not.toBe('octopus');
});
```

```
test('rejects to octopus', async () => {
  await expect(Promise.reject(new Error('octopus'))).rejects.toThrow('octopus');
});
```

- Matchers

- `.toBe(value)`
- `.toHaveBeenCalled()`
- `.toHaveBeenCalledTimes(number)`
- `.toHaveBeenCalledWith(arg1, arg2, ...)`
- `.toHaveBeenCalledWithLastCalledWith(arg1, arg2, ...)`
- `.toHaveBeenCalledWithNthCalledWith(nthCall, arg1, arg2,)`
- `.toHaveReturned()`
- `.toHaveReturnedTimes(number)`
- `.toHaveReturnedWith(value)`
- `.toHaveLastReturnedWith(value)`
- `.toHaveNthReturnedWith(nthCall, value)`
- `.toHaveLength(number)`



<https://jestjs.io/pt-BR/docs/expect>

Como escrever testes

Vamos praticar?



1. Voltar ao projeto clonado https://github.com/julialuiza/myapp_test
2. Criar arquivo de teste para as funções do arquivo `src/utils/validations.js`
 - a. Organizar da forma que preferir
 - i. em arquivo único ou separado para cada função;
 - ii. utilizando `describes` e `it`, somente `it`, mais de um `describe`, etc.
3. Considerando os exemplos mostrados e a documentação de matchers do Jest, criar testes unitários para cada função
4. Identificar, por meio dos testes, bugs existentes nas funções, e corrigi-las;
5. Obter os testes passando 100% .

Como escrever testes com Mock

Mock:

- Às vezes, durante a construção de um teste unitário, precisamos simular o comportamento de serviços dos quais nosso teste dependem, mas que **não são** o foco do teste em si.

Exemplo:

- Considerando uma função *verificaBairro(CEP: String)* que recebe um CEP, consulta os detalhes de endereço utilizando a API do viaCEP e verifica se está na lista de bairros atendidos pela loja;
 - Nesse caso, o que verdadeiramente interessa para nosso teste é se nossa função está conferindo corretamente a lista de bairros autorizados e retornando true ou false
 - Mas para isso, precisamos obter as informações de retorno da API do viaCEP;
- E agora?



Como escrever testes com Mock

Mock:

- Às vezes, durante a construção de um teste unitário, precisamos simular o comportamento de serviços dos quais nosso teste dependem, mas que **não são** o foco do teste em si.

Exemplo:

- Considerando uma função *verificaBairro(CEP: String)* que recebe um CEP, consulta os detalhes de endereço utilizando a API do viaCEP e verifica se está na lista de bairros atendidos pela loja;
- Nesse caso, o que verdadeiramente interessa para nosso teste é se nossa função está conferindo corretamente a lista de bairros autorizados e retornando true ou false
- Mas para isso, precisamos obter as informações de retorno da API do viaCEP;
- E agora?



Funções de Simulação

Funções de simulação (mocks em inglês) permitem que você teste os links entre códigos, apagando a implementação real de uma função, capturando chamadas para a função (e os parâmetros passados nessas chamadas), capturar instâncias do construtor de funções quando instanciado com `new`, e permitindo configuração em tempo de teste de valores de retorno.

Como escrever testes com Mock

```
export function forEach(items, callback) {
  for (let index = 0; index < items.length; index++) {
    callback(items[index]);
  }
}

const forEach = require('./forEach');

const mockCallback = jest.fn(x => 42 + x);

test('forEach mock function', () => {
  forEach([0, 1], mockCallback);

  // The mock function was called twice
  expect(mockCallback.mock.calls).toHaveLength(2);
}
```

Como escrever testes com Mock

```
import axios from 'axios';

class Users {
  static all() {
    return axios.get('/users.json').then(resp => resp.data);
  }
}

export default Users;
```

users.test.js

<https://jestjs.io/pt-BR/docs/mock-functions#simulando-m%C3%B3dulos>

```
import axios from 'axios';
import Users from './users';

jest.mock('axios');

test('deve buscar os usuários', () => {
  const users = [{name: 'Bob'}];
  const resp = {data: users};
  axios.get.mockResolvedValue(resp);

  // ou você poderá usar o código abaixo dependendo do seu caso de uso:
  // axios.get.mockImplementation(() => Promise.resolve(resp))

  return Users.all().then(data => expect(data).toEqual(users));
});
```

Como escrever testes com Mock

Vamos praticar?



1. Voltar ao projeto clonado https://github.com/julialuiza/myapp_test
2. Criar arquivo de teste para a função do arquivo `src/utils/cep.js` utilizando Mocks do Jest para simular a chamada ao módulo `node-correios`.
3. Dicas:
 - a. É possível utilizar **mockImplementation** do Jest para sobreescriver a função `consultaCEP()` do `node-correios`;
 - b. Por ser uma função assíncrona, é necessário considerar que o mock precisa retornar `promises`;
 - c. Da mesma forma, no `expect()` dos testes, é necessário considerar que as `promises` talvez ainda precisem ser resolvidas (`rejects/resolves`);
 - d. Consultar a implementação da função `consultaCEP()` original:
<https://github.com/vitorleal/node-correios/blob/master/lib/correios.js#L59>

Cobertura de código

COBERTURA DE CÓDIGO

Crie relatórios de cobertura de código facilmente usando `--coverage`. Sem necessidade adicional de configurações ou bibliotecas! Jest consegue coletar informações de cobertura de código de projetos inteiros, incluindo arquivos não testados.

- Pode ser incorporado em pipelines de CI/CD para manter uma cobertura mínima de código;
- Geralmente, usa-se a porcentagem mínima por volta de 80%, que é considerada boa.
- Para NPM:
 - `npm jest -- --coverage`

```
~/d/p/j/j/e/timer $ yarn jest --coverage
yarn run v1.12.3
$ /Users/ortatherox/dev/projects/jest/jest/examples/timer/node_modules/.bin/jest --coverage
PASS __tests__/timer_game.test.js
PASS __tests__/infinite_timer_game.test.js
-----|-----|-----|-----|-----|-----|
File      | %Stmts | %Branch | %Funcs | %Lines | Uncovered Line #s |
-----|-----|-----|-----|-----|-----|
All files |    87.5 |     100 |      80 |    87.5 |
infiniteTimerGame.js |    80 |     100 |  66.67 |     80 |      12 |
timerGame.js |   100 |     100 |    100 |    100 |
-----|-----|-----|-----|-----|-----|
Test Suites: 2 passed, 2 total
Tests:       4 passed, 4 total
Snapshots:  0 total
Time:        0.878s, estimated 1s
Ran all test suites.
```

Boas práticas em testes unitários

- **Nomear bem os testes:** é essencial nomear de forma semântica os blocos de testes e os testes em si; dessa forma, quando algum teste falhar será fácil identificar qual parte do código está com problemas.
- **Rápido:** não é incomum projetos maduros terem milhares de testes de unidade. A execução dos testes de unidade deve demorar pouco tempo.
- **Isolado:** testes de unidade são autônomos, podem ser executados em isolamento e geralmente não têm dependências em nenhum fator externo, como um sistema de arquivos ou o banco de dados. Caso seja necessário resolver dependências externas, utilize recursos como mock.
- **Repetível:** a execução de um teste de unidade deve ser consistente com seus resultados, ou seja, sempre retornará o mesmo resultado se você não alterar nada entre execuções.
- **Não Ignore os testes.**

<https://blog.mandic.com.br/artigos/10-dicas-para-escrever-bons-testes-de-unidade/>

<https://learn.microsoft.com/pt-br/dotnet/core/testing/unit-testing-best-practices>

Cronograma: Aula 01



ATENÇÃO

Dúvidas?

Visão geral

- Por que é importante testar?
- Diferentes tipos de teste;
- Ferramentas disponíveis.



Jest

- O que é e vantagens;
- Instalação utilizando npm;
- Configurações adicionais.



Testes no back-end

- Como escrever testes;
- Prática com testes para o back-end do projeto em andamento;
- Cobertura de código;
- Boas práticas.





Obrigada! =)

jlslc@icomp.ufam.edu.br