

Introdução à Automação de Testes com Jest

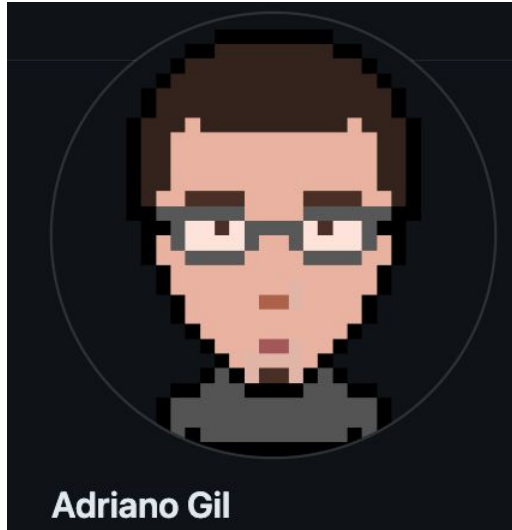
Adriano Gil

adrianozil.github.io

Agenda

- O que é Jest?
- O que são testes automatizados?
- Tipos de testes
- Testes de integração
- Testes unitários
- Mocks
- Cobertura de testes
- TDD

Quem sou eu?



- +10 anos trabalhando com desenvolvimento e pesquisa
- Mobile
- VR / AR
- Machine Learning
- Games
- Papers publicados sobre Automação de Testes
 - SBGAMES, SIBGRAPI, HCII

Quem sou eu?



- Backend NodeJS
- APIs de compra (Stripe, Paypal, ...)
- NPM
- Jest

O que é Jest?

- Framework de testes JavaScript mantido pelo Facebook
- Integrado com Ecosystema Javascript
 - NodeJs / React / Angular ...
- <https://jestjs.io/docs/getting-started>



O que é Jest?

- Test Runner
- Responsável por:
 - Encontrar os arquivos de testes (*.test.js ou *.spec.js)
 - Rodar os testes
 - Determinar se os testes passaram ou falharam

O que é Jest?

- Assertions
- expect(X).toBe(Y)
 - Checa se X === Y

```
test("a soma de dois numeros", () => {  
  expect(1 + 2).toBe(3);  
});
```

Como rodar os testes usando Jest?

- npm install jest --save-dev
- npm test
 - Alias: npm run test

```
└─ npm test

> talk-intro-jest-01@1.0.0 test
> jest

PASS tests/fast-inverse-square-root.test.js
PASS tests/numbers.test.js
PASS tests/location.test.js
PASS tests/simple.test.js

Test Suites: 4 passed, 4 total
Tests:       4 passed, 4 total
Snapshots:  0 total
Time:        0.84 s, estimated 1 s
Ran all test suites.
```


Como rodar os testes usando Jest?

- npm test
 - Alias para npm run test
 - Deve existir uma entrada "test" no package.json com o comando a ser rodado

```
"version": "1.0.0",  
"description": "",  
"main": "index.js",  
  Debug  
"scripts": {  
  "test": "jest"  
},  
"keywords": []
```

Como rodar os testes usando Jest?

- npm test
 - Alias para npm run test
 - Deve existir uma entrada "test" no package.json com o comando a ser rodado

```
function npm-test()
{
  # Use the correct version of Node.js for the project
  nvm use

  # Install dependencies
  npm install

  # Run the tests
  npm test
}
alias ntest="npm-test"
```

https://github.com/adrianoqil/nodeutils/blob/main/node_test.sh#L2

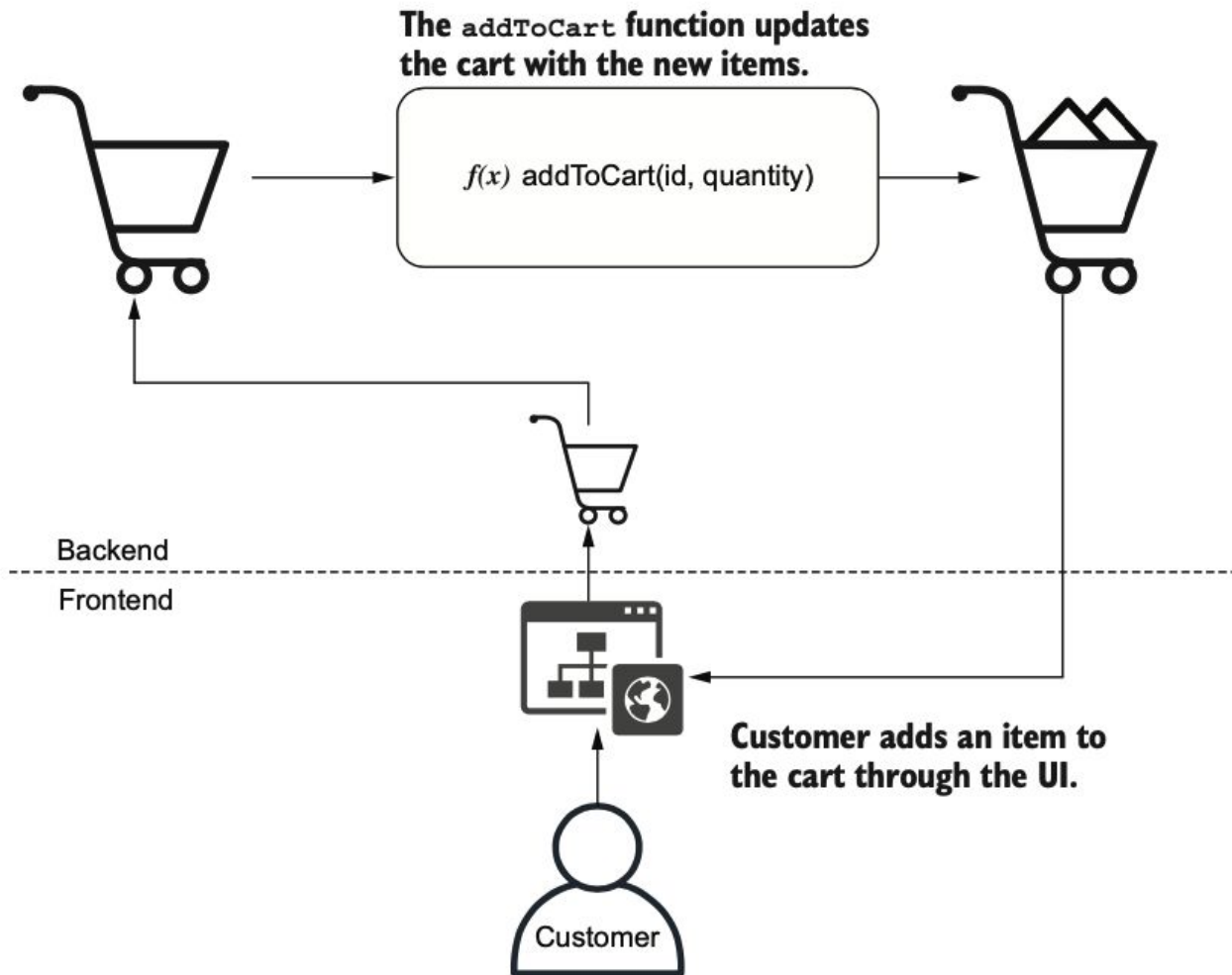
Watch mode

- `--watch`
 - Monitora arquivos específicos e roda os testes associados quando há modificações.
- `--watchAll`
 - Monitora todos os arquivos no projeto e reexecuta todos os testes quando qualquer arquivo é modificado.

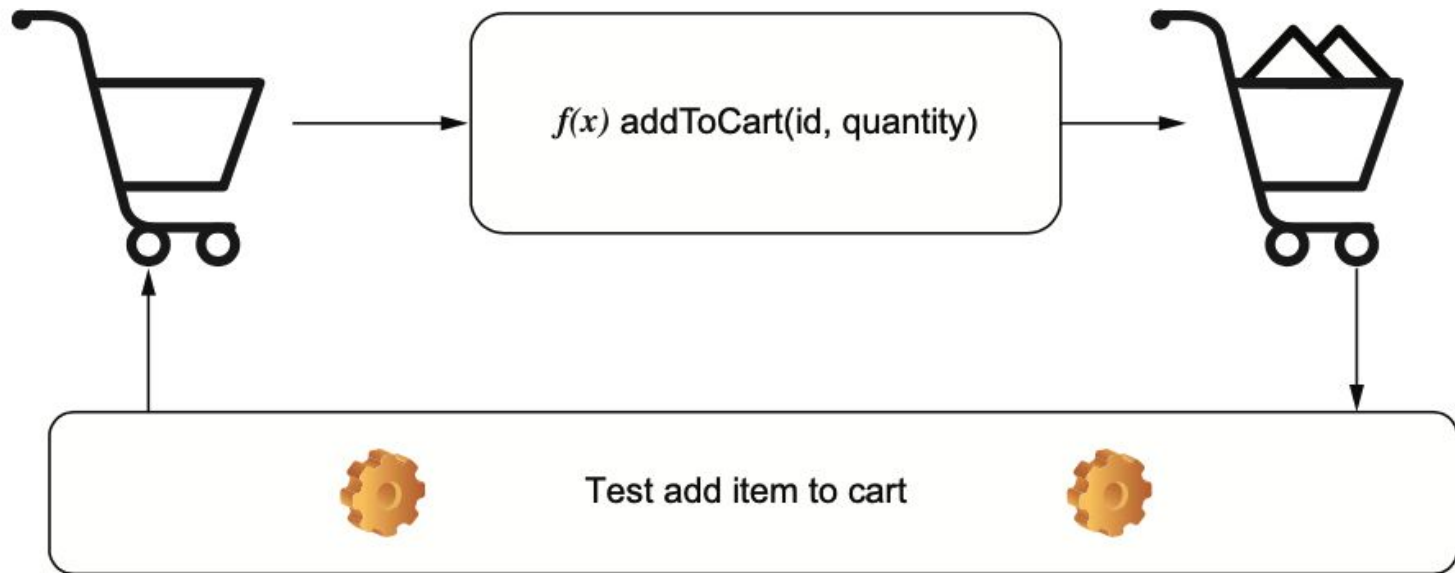
O que são testes automatizados?

- Testes automatizados são programas que automatizam a tarefa de testar seu software.
- Eles interagem com sua aplicação para realizar ações e comparar o resultado real com a saída esperada que você definiu previamente.





The `addToCart` function updates the cart with the new items.



An automated test uses the `addToCart` function to update the cart and checks the cart's final state.

The `addToCart` function updates the cart with the new items.

$f(x)$ `addToCart(id, quantity)`



Create an empty cart.

Try adding 2 macaroons to the cart.

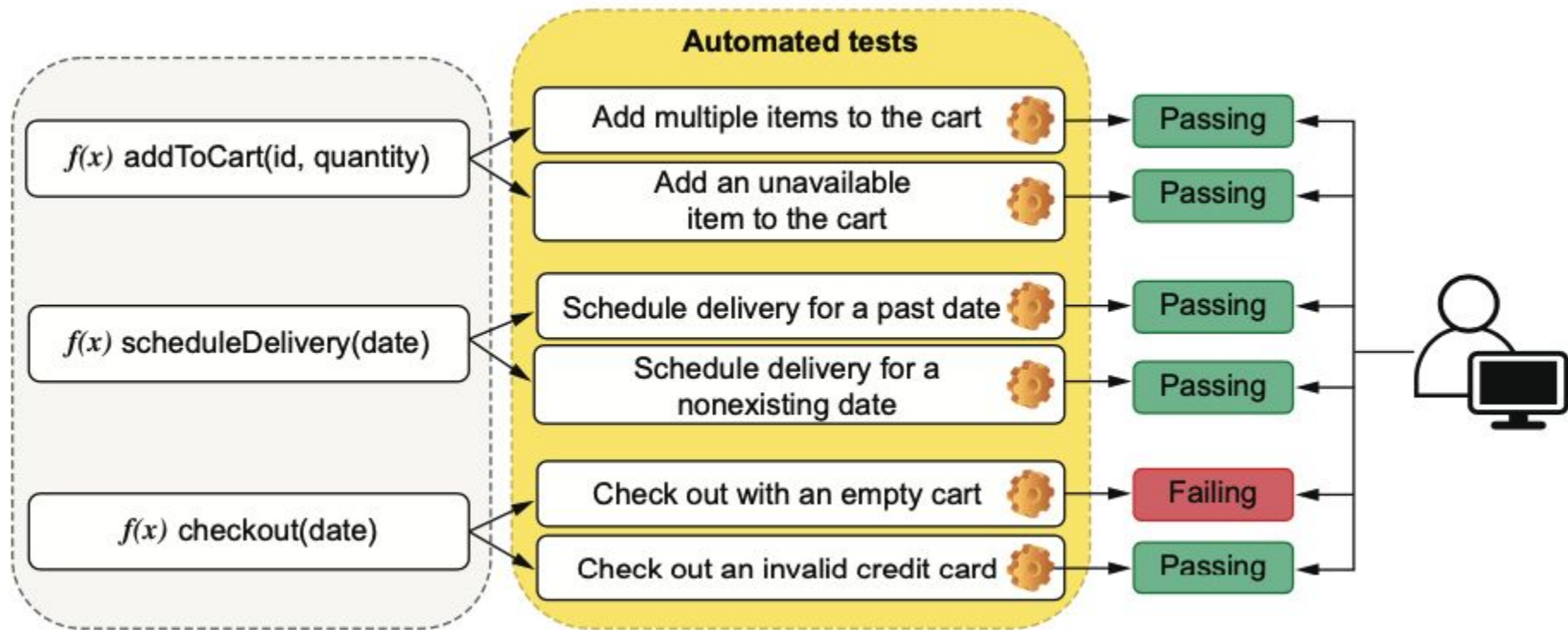
Check if the cart contains 2 macaroons.

It's possible to add multiple macaroons to the cart. ✓



Test
add item to cart





Benefícios dos Testes Automatizados

- Consistência e repetibilidade.
- Economia de tempo e recursos.
- Detecção precoce de bugs.
- Melhorias na qualidade do software.



Benefícios dos Testes Automatizados

(COLLINS; DE LUCENA, 2012)

- Reduz o tempo gasto em testes manuais e aumenta a cobertura dos testes.
- Foco em automatizar testes críticos primeiro e expandir conforme necessário.

Software Test Automation Practices in Agile Development Environment: An Industry Experience Report

Eliane Figueiredo Collins
*Nokia Institute of Technology
Manaus, AM, Brazil
eliane.collins@indt.org.br*

Vicente Ferreira de Lucena Jr.
*Federal University of Amazonas
Manaus, AM, Brazil
Vicente@ufam.edu.br*

Benefícios dos Testes Automatizados

(WILLIAMS; KUDRJAVETS; NAGAPPAN, 2009)

Após um período de um ano implementando testes unitários perceberam:

- uma redução de 20,9% nos defeitos de teste
- 30% mais tempo de desenvolvimento
- redução dos defeitos encontrados pelos clientes

20th International Symposium on Software Reliability Engineering

On the Effectiveness of Unit Test Automation at Microsoft

Laurie Williams¹, Gunnar Kudrjavets², and Nachiappan Nagappan²

¹Department of Computer Science, North Carolina State University

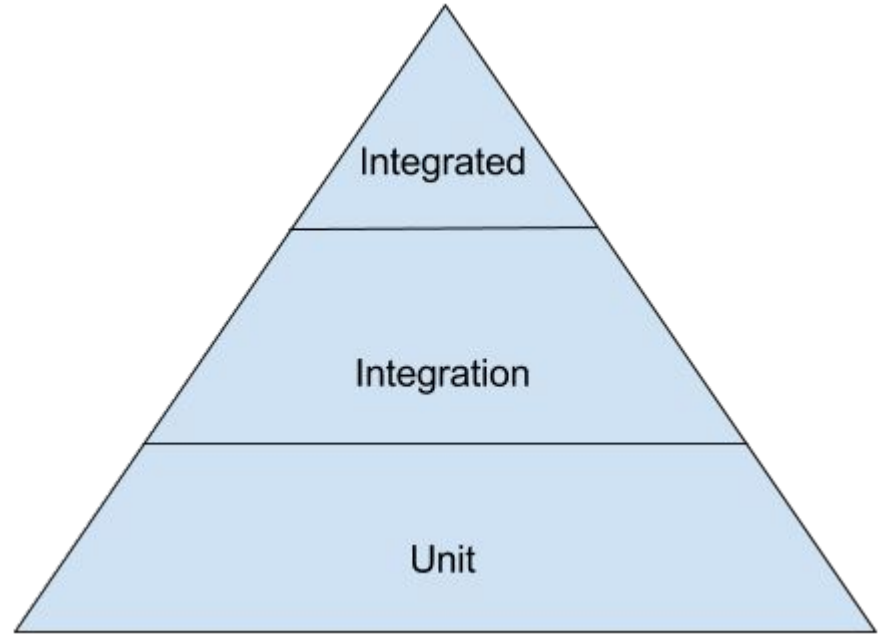
williams@ncsu.edu

²Microsoft Corporation

{gunnarku, nachin}@microsoft.com

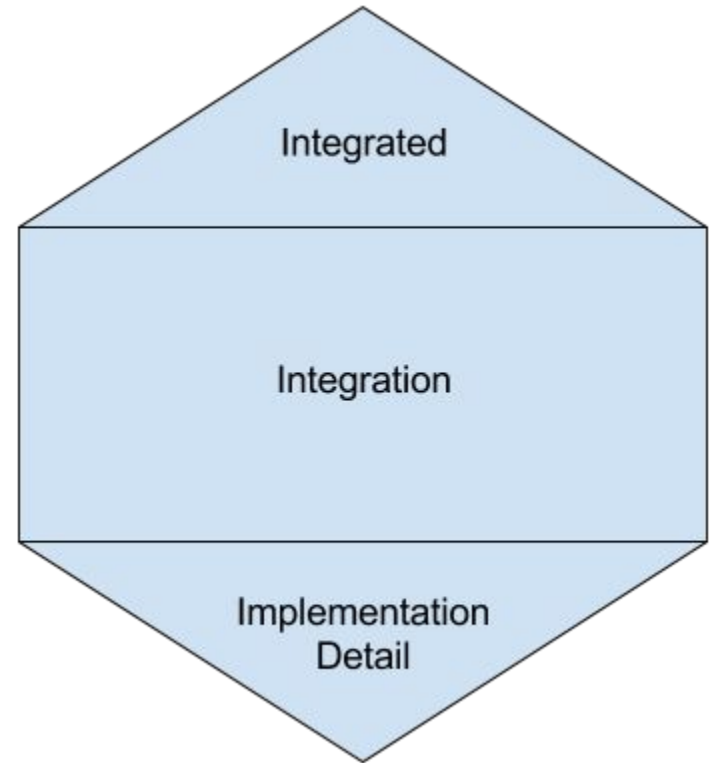
Tipos de Testes Automatizados

- Testes unitários.
- Testes de integração.
- Testes de ponta a ponta (end-to-end).



Tipos de Testes Automatizados

- Testes unitários.
- Testes de integração.
- Testes de ponta a ponta (end-to-end).



Modelo Spotify

Testes de Integração

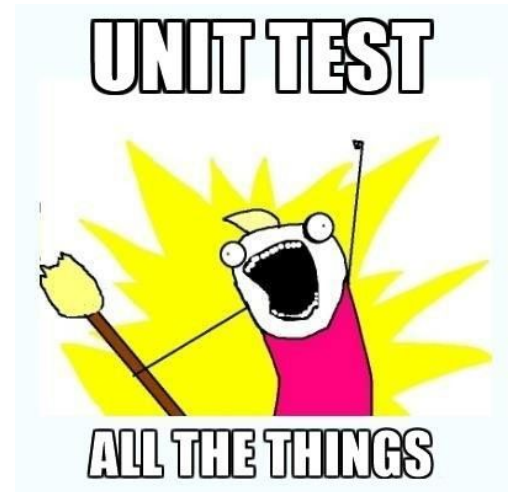
Testes de integração verificam a interação entre diferentes componentes ou módulos de um sistema para garantir que eles funcionem juntos como esperado.

Tem por objetivo:

- Verificar se a comunicação entre componentes é correta.
- Assegurar que os componentes integrados atendam aos requisitos de negócio

Testes Unitários

- Testam a menor unidade de código
 - como funções ou métodos individuais.
- Facilitar a detecção de bugs logo no início do desenvolvimento.
- Utilizar mocks e stubs para isolar a unidade de código.
- Triangulação de erros



Exemplo de teste unitário de uma função

```
function getLocation() {  
  return 'Vault 76';  
}  
  
test('retorna a localização correta', () => {  
  expect(getLocation()).toBe('Vault 76');  
});
```


Exemplo de teste unitário de uma função

```
function getLotteryNumbers() {  
  return [4, 8, 15, 16, 23, 42];  
}  
  
test('returns the correct lottery numbers', () => {  
  expect(getLotteryNumbers()).toEqual([4, 8, 15, 16, 23, 42]);  
});
```

Exemplo de teste unitário de uma função

- Fast inverse square root



```
function Q_rsqrt(x) {  
    const threehalfs = 1.5;  
    const x2 = x * 0.5;  
    const magicNumber = 0x5f3759df;  
  
    // Convert the float to an integer using a trick.  
    const buf = new ArrayBuffer(4);  
    const f32 = new Float32Array(buf);  
    const u32 = new Uint32Array(buf);  
  
    f32[0] = x;  
    u32[0] = magicNumber - (u32[0] >> 1);  
  
    // Convert the integer back to a float.  
    const y = f32[0];  
  
    // Perform one iteration of Newton's method.  
    return y * (threehalfs - (x2 * y * y));  
}
```

Exemplo de teste unitário de uma função

- Fast inverse square root

```
test('fast inverse square root of 4', () => {  
  const number = 4.0;  
  const expected = 1 / Math.sqrt(number);  
  const result = Q_rsqrt(number);  
  expect(result).toBeCloseTo(expected);  
});
```

Exemplo de teste unitário de uma função

- Fast inverse square root

```
test.each([[1], [2], [4], [9], [16], [25], [36]])(  
  'fast inverse square root of %i',  
  (value) => {  
    expect(Q_rsqrt(value)).toBeCloseTo(1 / Math.sqrt(value), 2);  
  }  
);
```

Property-based testing

Exemplo de teste API Rest

- express
 - Lib para criação de APIs RESTful.
 - manipulação de URLs e endpoints
 - processa requisições e respostas



```
const express = require('express');
const app = express();
const port = 3000;

// Pikachu data
const pikachu = {
  name: 'Pikachu',
  type: 'Electric',
  abilities: ['Static', 'Lightning Rod'],
};

const pokemons = [
  pikachu
];

// Define a GET endpoint for Pikachu
app.get('/pokedex/', (req, res) => {
  res.json({ pokemons });
});

// Start the server
app.listen(port, () => {
  console.log(`Server is running at http://localhost:\${port}`);
});
```

Exemplo de teste API Rest

```
└─ curl -s http://localhost:3000/pokedex/ | jq
{
  "pokemons": [
    {
      "name": "Pikachu",
      "type": "Electric",
      "abilities": [
        "Static",
        "Lightning Rod"
      ]
    }
  ]
}
```

```
const express = require('express');
const app = express();
const port = 3000;

// Pikachu data
const pikachu = {
  name: 'Pikachu',
  type: 'Electric',
  abilities: ['Static', 'Lightning Rod'],
};

const pokemons = [
  pikachu
];

// Define a GET endpoint for Pikachu
app.get('/pokedex/', (req, res) => {
  res.json({ pokemons });
});

// Start the server
app.listen(port, () => {
  console.log(`Server is running at http://localhost:\${port}`);
});
```

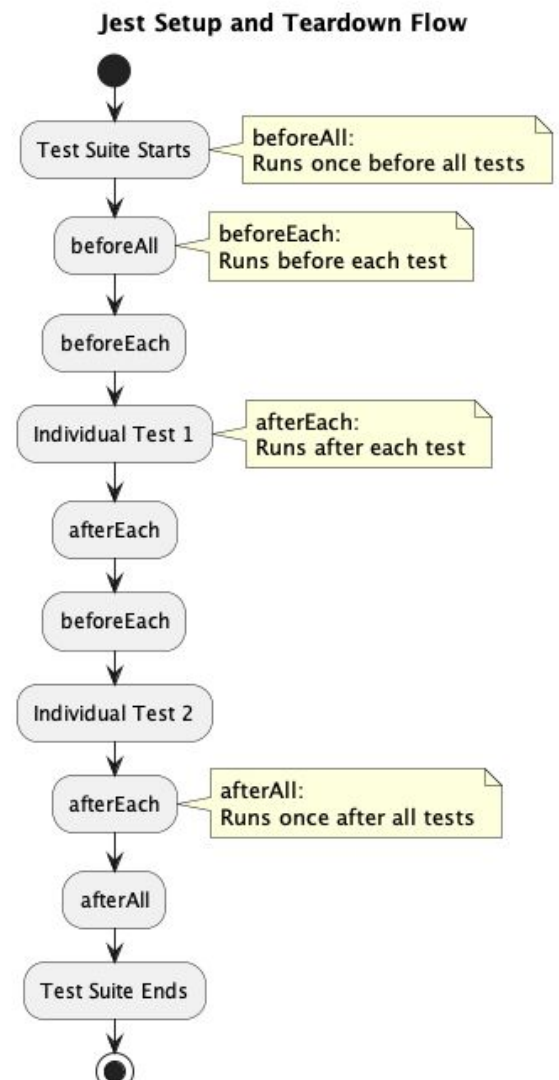
Exemplo de teste API Rest

- Verifica se a resposta da API contém os campos necessários.
- Utiliza a lib supertest para executar request
- toHaveProperty

```
test('Pokémon response contains required fields', async () => {  
  const response = await request(app).get('/pokedex/');  
  const pikachu = response.body.pokemons[0];  
  expect(pikachu).toHaveProperty('name', 'Pikachu');  
  expect(pikachu).toHaveProperty('type', 'Electric');  
});
```

Setup e Teardown

- Setup
 - beforeAll
 - beforeEach
- Teardown
 - afterAll
 - afterEach

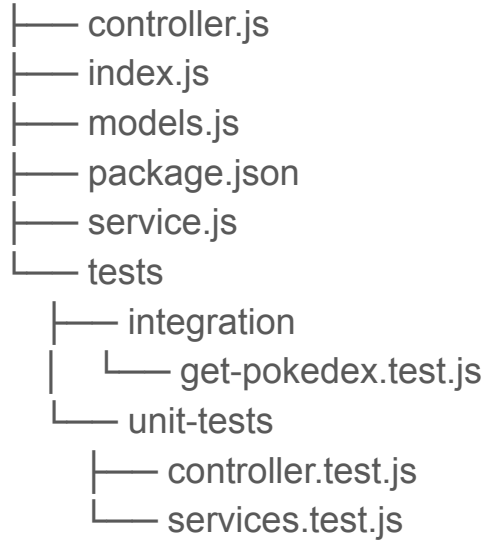


Mocks

- Funções mock no Jest permitem criar e controlar implementações "falsas" de funções, métodos ou módulos durante os testes.
- `jest.fn()` ou `jest.mock()`
- Monkey patch




Rest API + DB



Rest API + DB

- sequelize
 - ':memory'
 - DB SQLite em memória

 Pokemon
<ul style="list-style-type: none">○ int id○ String name○ String type○ JSON abilities

models.js

```
const { Sequelize, DataTypes } = require('sequelize');
const sequelize = new Sequelize({
  dialect: 'sqlite',
  storage: ':memory:'
});

const Pokemon = sequelize.define('Pokemon', {
});

const initDB = async () => {
  await sequelize.sync({ force: true });

  // Seed database with Pikachu
  await Pokemon.create({
    name: 'Pikachu',
    type: 'Electric',
    abilities: ['Static', 'Lightning Rod']
  });
};

module.exports = { Pokemon, initDB };
```

Rest API + DB

- Service que comunica com o ORM para obter os dados

service.js

```
const { Pokemon, initDB } = require('./models');

// Initialize the database and seed it with data
initDB().then(() => {
  console.log('Database initialized');
});

const getPokemons = async () => {
  return await Pokemon.findAll();
};

exports.getPokemons = getPokemons;
```

Rest API + DB

- Controller implementa o comportamento da rota

controller.js

```
const {getPokemons} = require('./service')

handleGetPokemon = async (req, res) => {
  var pokemons = await getPokemons();
  pokemons = pokemons.map(
    { name, type, abilities } => ({ name, type, abilities }));
  res.json({ pokemons });
};

exports.handleGetPokemon = handleGetPokemon;
```

Rest API + DB

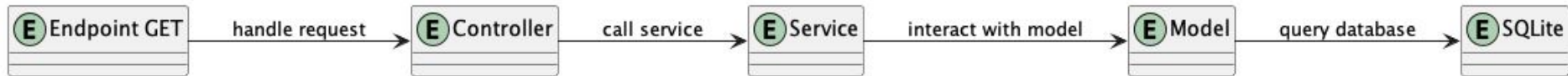
- Define endpoint implementado pelo controller

index.js

```
const express = require('express');
const { handleGetPokemon } = require('./controller');
const app = express();
const port = 3000;

// Define a GET endpoint for Pikachu
app.get('/pokedex/', handleGetPokemon);

// Start the server
app.listen(port, () => {
  console.log(`Server is running at http://localhost:${port}`);
});
```



Rest API + DB

```
└─ curl -s http://localhost:3000/pokedex/ | jq
{
  "pokemons": [
    {
      "name": "Pikachu",
      "type": "Electric",
      "abilities": [
        "Static",
        "Lightning Rod"
      ]
    }
  ]
}
```

index.js

```
const express = require('express');
const { handleGetPokemon } = require('./controller');
const app = express();
const port = 3000;

// Define a GET endpoint for Pikachu
app.get('/pokedex/', handleGetPokemon);

// Start the server
app.listen(port, () => {
  console.log(`Server is running at http://localhost:${port}`);
});
```



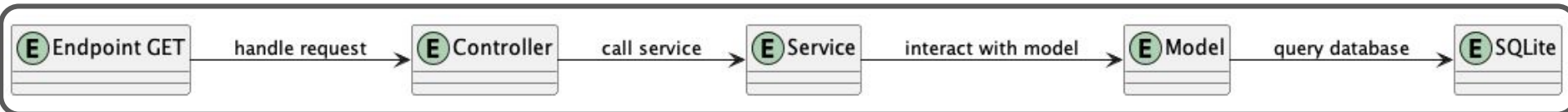
Rest API + DB

Teste de integração

integration/get-pokedex.test.js

```
// Initialize the database before running the tests
beforeAll(async () => {
  await initDB();
});

test('Pokémon response contains required fields', async () => {
  const response = await request(app).get('/pokedex/');
  const pikachu = response.body.pokemons[0];
  expect(pikachu).toHaveProperty('name', 'Pikachu');
  expect(pikachu).toHaveProperty('type', 'Electric');
});
```



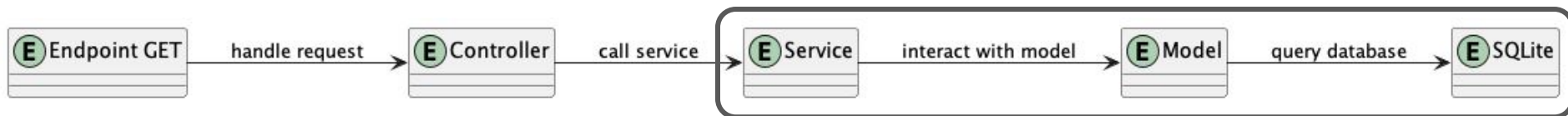
Rest API + DB Testes Unitários

unit-tests/services.test.js

```
const { getPokemons } = require('../..//service');
const { initDB } = require('../..//models');

describe('getPokemons', () => {
  beforeAll(async () => {
    await initDB();
  });

  test('retorna um Pikachu com as properties corretas', async () => {
    const pokemons = await getPokemons();
    const pikachu = pokemons.find(p => p.name === 'Pikachu');
    expect(pikachu).toBeDefined();
    expect(pikachu).toHaveProperty('name', 'Pikachu');
    expect(pikachu).toHaveProperty('type', 'Electric');
    expect(pikachu).toHaveProperty('abilities', ['Static', 'Lightning Rod']);
  });
});
```



```
describe('handleGetPokemon', () => {
  let req, res;

  beforeEach(() => {
    req = {};
    res = {
      json: jest.fn()
    };
  });

  test('should return pokemons with correct properties', async () => {
    // Arrange
    const mockPokemons = [
      { name: 'Pikachu', type: 'Electric', abilities: ['Static', 'Lightning Rod'] }
    ];
    getPokemons.mockResolvedValue(mockPokemons);

    // Act
    await handleGetPokemon(req, res);

    // Assert
    expect(getPokemons).toHaveBeenCalled();
    expect(res.json).toHaveBeenCalledWith({
      pokemons: [
        { name: 'Pikachu', type: 'Electric', abilities: ['Static', 'Lightning Rod'] }
      ]
    });
  });
});
```

Rest API + DB

Testes Unitários

- mockResolvedValue
- toHaveBeenCalledWith



Erros comuns

- Funções Async devem ser mockadas por equivalentes async
 - `mockReturnValue` x `mockResolvedValue`
- `jest.resetAllMocks()` x `jest.restoreAllMocks()`
- Descrições de teste com o mesmo nome no mesmo arquivo

Global hooks

- globalSetup
- globalTeardown

```
// jest.config.js
module.exports = {
  ...
  // Other Jest configuration options...
  globalSetup: './globalSetup.js',
  globalTeardown: './globalTeardown.js'
};
```

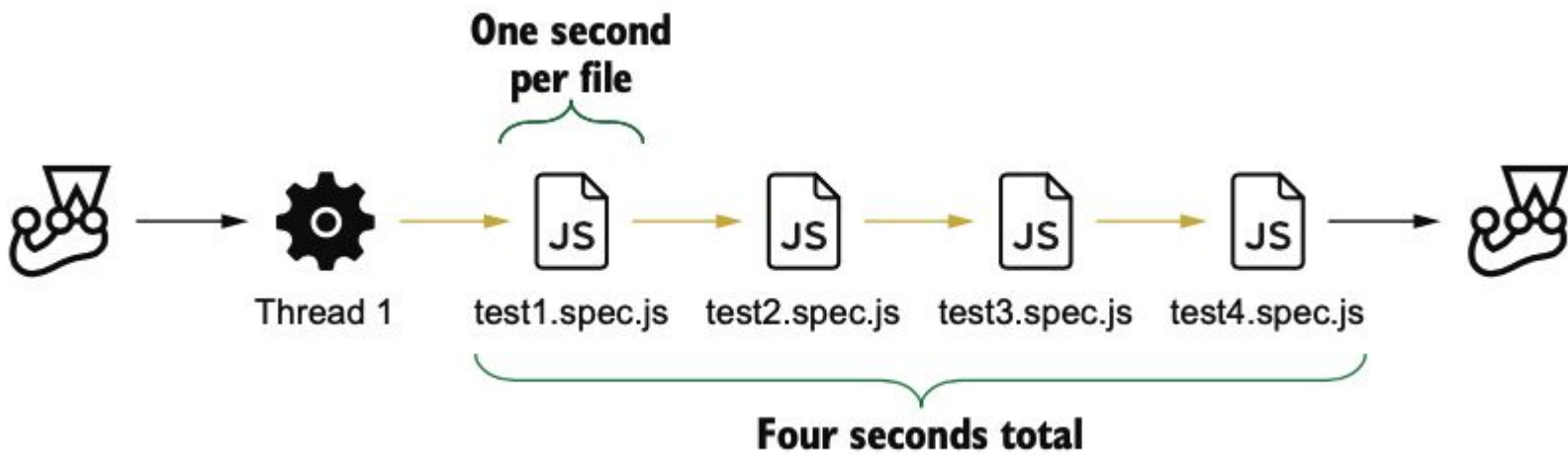
Processamento paralelo

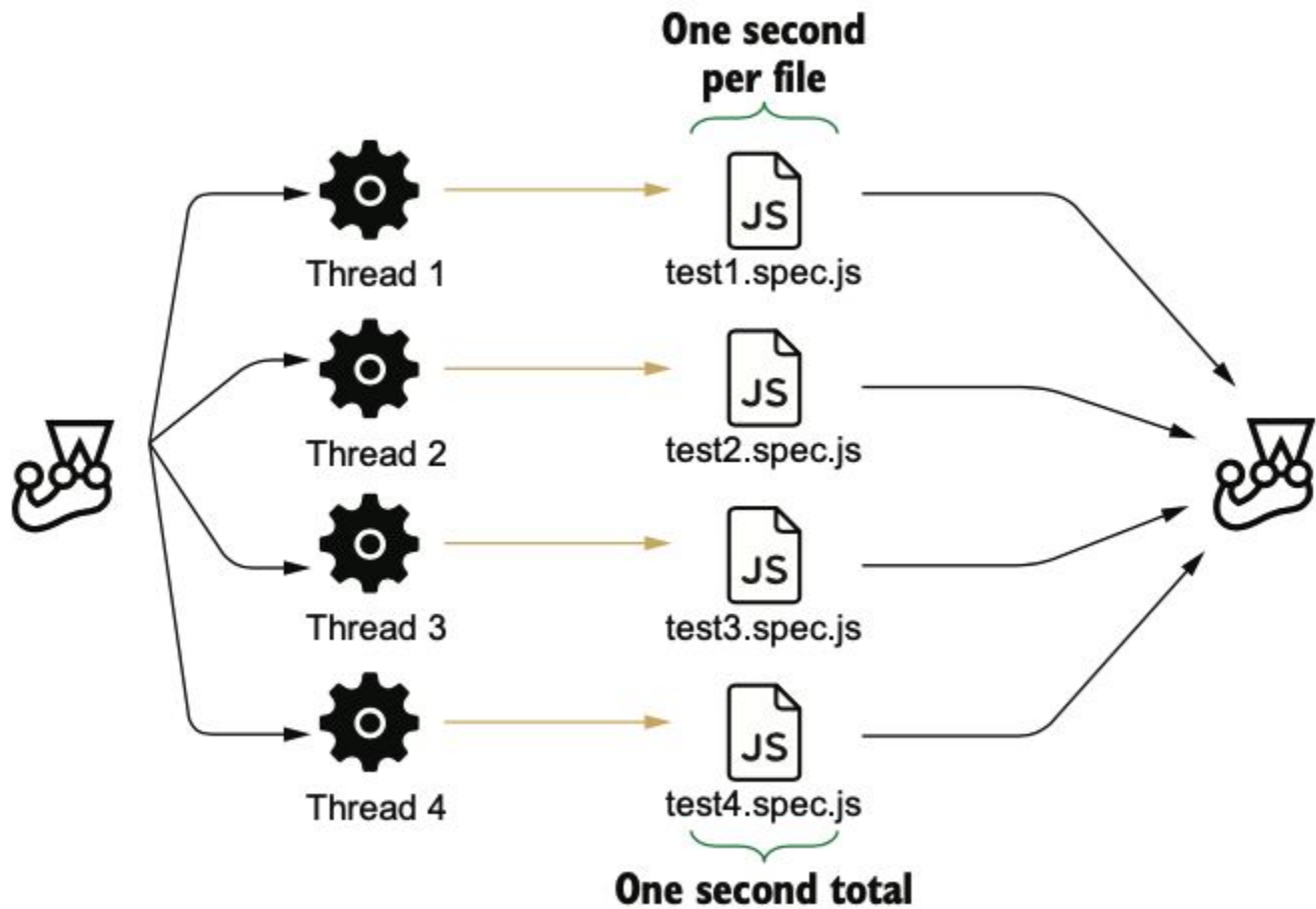
`--runInBand`

- executa todos os testes sequencialmente

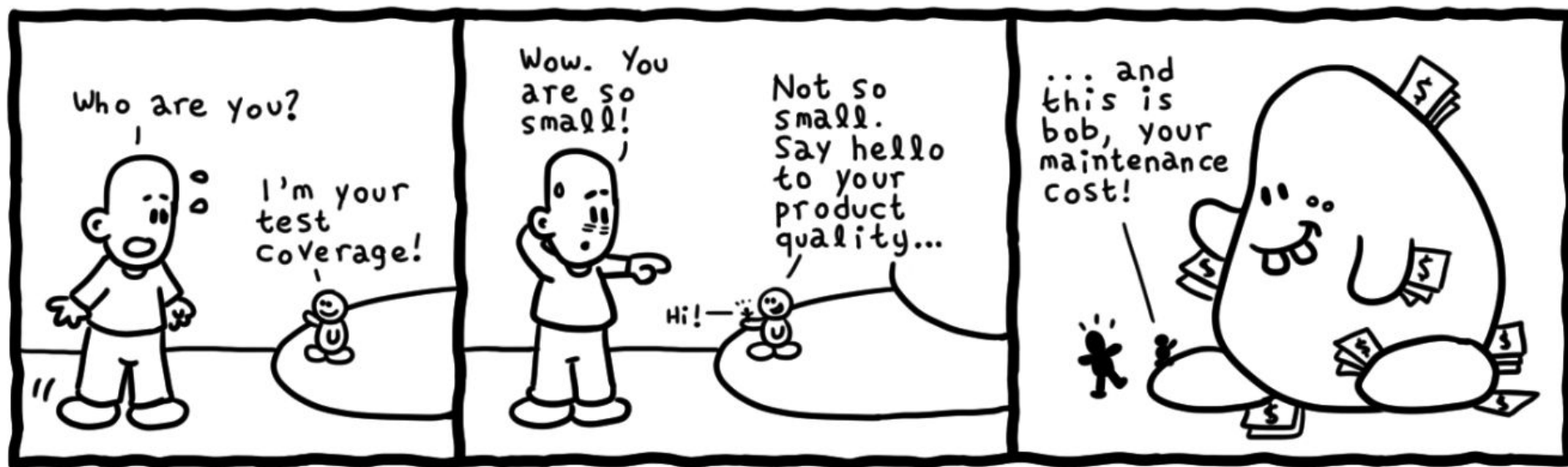
`--maxWorkers`

- especificar o número máximo de processos para executar os testes





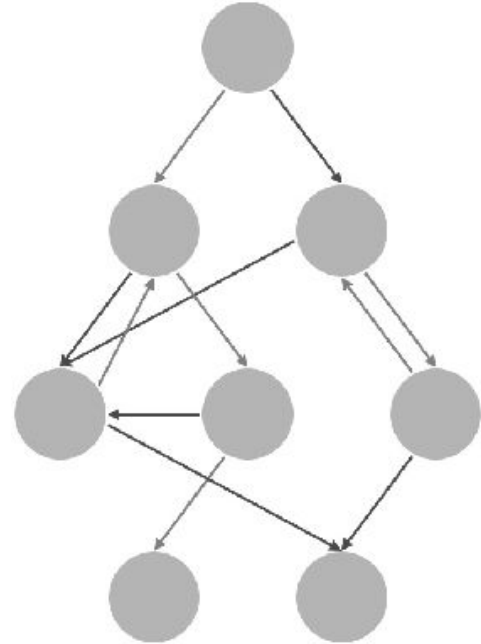
Cobertura de Testes



Daniel Stori {turnoff.us}

Cobertura de Testes

- Medida da quantidade de código que é executada durante os testes.
- Indica a eficácia dos testes em cobrir diferentes caminhos e condições do código.



Cobertura de Testes

jest --coverage

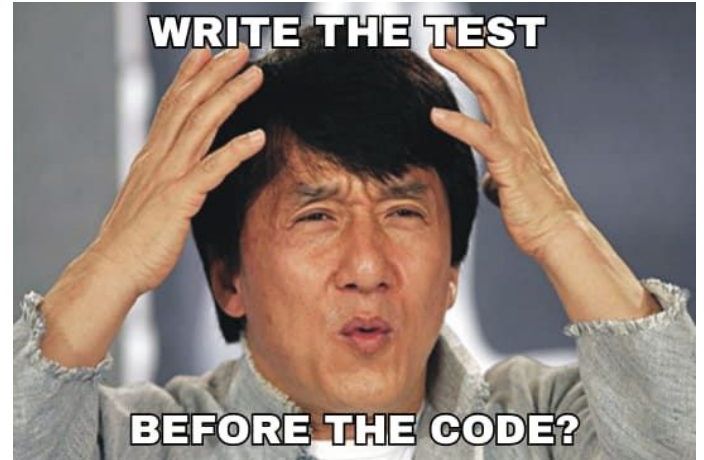
- **Cobertura de Linhas:** Percentual de linhas de código executadas pelos testes.
- **Cobertura de Funções:** Percentual de funções/métodos executados pelos testes.
- **Cobertura de Ramos:** Percentual de ramos de decisão (if, switch, etc.) executados pelos testes.
- **Cobertura de Declarações:** Percentual de declarações executadas pelos testes.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	100	100	100	100	
controller.js	100	100	100	100	
http-server.js	100	100	100	100	
models.js	100	100	100	100	
service.js	100	100	100	100	

Test Suites: 3 passed, 3 total
Tests: 4 passed, 4 total
Snapshots: 0 total
Time: 2.052 s
Ran all test suites.

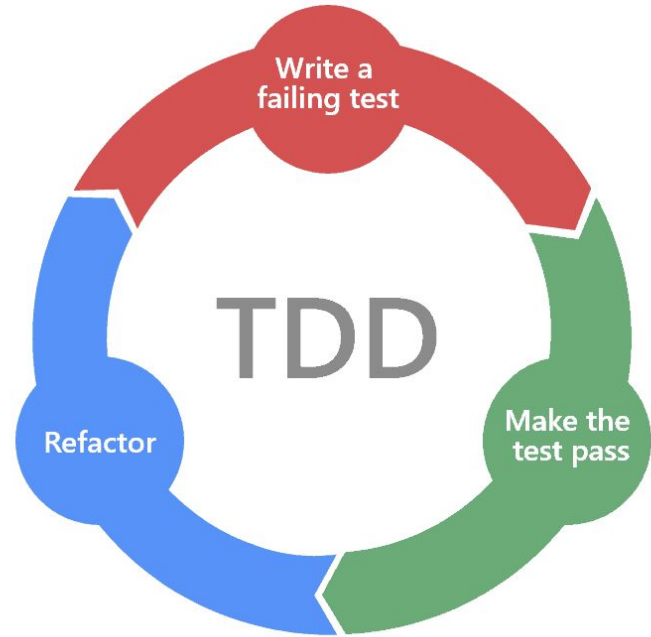
TDD

- Test-Driven Development
- prática de desenvolvimento de software onde os testes são escritos antes do código funcional



TDD

1. Escrever um teste: Criar um teste que falha.
2. Escrever código: Desenvolver o código mínimo necessário para passar no teste.
3. Refatorar: Melhorar o código mantendo todos os testes passando.



Conclusão

- Jest agiliza o desenvolvimento de testes
- TDD promove um código mais limpo e bem estruturado.
- TDD + jest --watch



Referências

JEST. Jest: Delightful JavaScript Testing. <https://jestjs.io/>.

SPOTIFY ENGINEERING. Testing of Microservices. 2018.
<https://engineering.atspotify.com/2018/01/testing-of-microservices/>.

Referências

COLLINS, Eliane Figueiredo; DE LUCENA, Vicente Ferreira. **Software test automation practices in agile development environment: An industry experience report**. In: INTERNATIONAL WORKSHOP ON AUTOMATION OF SOFTWARE TEST (AST), 7., 2012. Anais [...]. [S.l.]: IEEE, 2012. p. 57-63.

WILLIAMS, Laurie; KUDRJAVETS, Gunnar; NAGAPPAN, Nachiappan. **On the effectiveness of unit test automation at Microsoft**. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY ENGINEERING, 20., 2009. Anais [...]. [S.l.]: IEEE, 2009. p. 81-89.

DA COSTA, Lucas Fernandes. **Testing JavaScript Applications**. 2021. New York: Simon and Schuster.

SINGH, Santosh Kumar. **JavaScript Testing with Jest: A TDD Approach (English Edition)**. 2021.

Introdução à Automação de Testes com Jest

Adriano Gil

adrianozil.github.io