# Estrutura do Projeto

car-rental-system/

├── src/

│ ├── car.js # Lógica para a classe Carro

│ ├── rental.js # Lógica para o sistema de locação

│ └── utils.js # Funções utilitárias

├── tests/

│ ├── car.test.js # Testes unitários para Carro

│ └── rental.test.js # Testes unitários para Rental

├── package.json # Metadados do projeto e dependências

├── jest.config.js # Configuração do Jest

└── stryker.conf.js # Configuração do Stryke

# Inicialização do Projeto

1. Criação das Pastas necessárias

mkdir car-rental-system

cd car-rental-system

1. Inicialização

npm init -y

1. Instalação de Dependências

npm install jest --save-dev

npm install stryker stryker-api stryker-cli --save-dev

# Implementação das Classes e Funções

**File: src/car.js**

class Carro {

constructor(id, modelo, marca, ano, disponivel = true) {

this.id = id;

this.modelo = modelo;

this.marca = marca;

this.ano = ano;

this.disponivel = disponivel;

}

alugar() {

if (!this.disponivel) {

throw new Error("Carro já está alugado.");

}

this.disponivel = false;

}

devolver() {

if (this.disponivel) {

throw new Error("Carro está disponível.");

}

this.disponivel = true;

}

}

module.exports = Carro;

**File: src/rental.js**

class RentalSystem {

constructor() {

this.carros = [];

}

adicionarCarro(carro) {

this.carros.push(carro);

}

removerCarro(id) {

this.carros = this.carros.filter(carro => carro.id !== id);

}

listarCarros() {

return this.carros;

}

alugarCarro(id) {

const carro = this.carros.find(carro => carro.id === id);

if (!carro) {

throw new Error("Carro não encontrado.");

}

carro.alugar();

}

devolverCarro(id) {

const carro = this.carros.find(carro => carro.id === id);

if (!carro) {

throw new Error("Carro não encontrado.");

}

carro.devolver();

}

}

module.exports = RentalSystem;

**File: src/utils.js**

# Criação dos Testes Unitários com Jest

**File: tests/car.test.js**

const Carro = require('../src/car');

describe('Carro', () => {

let carro;

beforeEach(() => {

carro = new Carro(1, 'ModeloX', 'MarcaY', 2020);

});

it('deve criar um carro com as propriedades corretas', () => {

console.log('Teste: deve criar um carro com as propriedades corretas');

expect(carro.id).toBe(1);

expect(carro.modelo).toBe('ModeloX');

expect(carro.marca).toBe('MarcaY');

expect(carro.ano).toBe(2020);

expect(carro.disponivel).toBe(true);

console.log('Carro criado:', carro);

});

it('deve alugar um carro disponível', () => {

console.log('Teste: deve alugar um carro disponível');

expect(carro.disponivel).toBe(true);

console.log('Disponibilidade antes:', carro.disponivel);

carro.alugar();

expect(carro.disponivel).toBe(false);

console.log('Disponibilidade depois:', carro.disponivel);

console.log('Carro alugado:', carro);

});

it('não deve alugar um carro já alugado', () => {

console.log('Teste: não deve alugar um carro já alugado');

carro.disponivel = false;

expect(() => carro.alugar()).toThrow("Carro já está alugado.");

});

it('deve devolver um carro alugado', () => {

console.log('Teste: deve devolver um carro alugado');

carro.disponivel = false;

expect(carro.disponivel).toBe(false);

console.log('Disponibilidade antes:', carro.disponivel);

carro.devolver();

expect(carro.disponivel).toBe(true);

console.log('Disponibilidade depois:', carro.disponivel);

console.log('Carro devolvido:', carro);

});

it('não deve devolver um carro já disponível', () => {

console.log('Teste: não deve devolver um carro já disponível');

expect(() => carro.devolver()).toThrow("Carro está disponível.");

});

});

**File: tests/rental.test.js**

const RentalSystem = require('../src/rental');

const Carro = require('../src/car');

describe('RentalSystem', () => {

let rentalSystem;

let carro1, carro2;

beforeEach(() => {

rentalSystem = new RentalSystem();

carro1 = new Carro(1, 'ModeloX', 'MarcaY', 2020);

carro2 = new Carro(2, 'ModeloZ', 'MarcaW', 2022);

rentalSystem.adicionarCarro(carro1);

rentalSystem.adicionarCarro(carro2);

});

it('deve adicionar um carro', () => {

const carro3 = new Carro(3, 'ModeloA', 'MarcaB', 2023);

rentalSystem.adicionarCarro(carro3);

expect(rentalSystem.listarCarros().length).toBe(3);

});

it('deve remover um carro', () => {

rentalSystem.removerCarro(1);

expect(rentalSystem.listarCarros().length).toBe(1);

expect(rentalSystem.listarCarros()[0].id).toBe(2);

});

it('deve listar todos os carros', () => {

const carros = rentalSystem.listarCarros();

expect(carros.length).toBe(2);

expect(carros[0].id).toBe(1);

expect(carros[0].modelo).toBe('ModeloX');

expect(carros[0].marca).toBe('MarcaY');

expect(carros[0].ano).toBe(2020);

expect(carros[0].disponivel).toBe(true);

expect(carros[1].id).toBe(2);

expect(carros[1].modelo).toBe('ModeloZ');

expect(carros[1].marca).toBe('MarcaW');

expect(carros[1].ano).toBe(2022);

expect(carros[1].disponivel).toBe(true);

});

it('deve alugar um carro disponível', () => {

rentalSystem.alugarCarro(1);

expect(carro1.disponivel).toBe(false);

});

it('não deve alugar um carro não encontrado', () => {

expect(() => rentalSystem.alugarCarro(3)).toThrow("Carro não encontrado.");

});

it('deve devolver um carro alugado', () => {

rentalSystem.alugarCarro(1);

rentalSystem.devolverCarro(1);

expect(carro1.disponivel).toBe(true);

});

it('não deve devolver um carro não encontrado', () => {

expect(() => rentalSystem.devolverCarro(3)).toThrow("Carro não encontrado.");

});

});

# Configuração do JEST

**jest.config.js**

module.exports = {

testEnvironment: 'node',

testMatch: ['\*\*/tests/\*\*/\*.test.js'],

collectCoverage: true,

coverageDirectory: 'coverage',

coverageReporters: ['text', 'lcov'],

};

# Execução dos Testes Unitários

**File: package.json**

{

  "name": "car-rental-system",

  "version": "1.0.0",

  "main": "index.js",

  "scripts": {

    "test": "jest",

    "mutation": "stryker run"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "description": "",

  "devDependencies": {

    "jest": "^29.7.0",

    "stryker": "^0.2.1",

    "stryker-api": "^0.9.0",

    "stryker-cli": "^1.0.2"

  }

}

# Configuração do Stryker

**File:stryker.conf.js**

module.exports = {

    mutator: 'javascript',

    packageManager: 'npm',

    reporters: ['html', 'clear-text', 'progress'],

    testRunner: 'jest',

    transpilers: [],

    coverageAnalysis: 'off',

    mutate: [

      'src/\*\*/\*.js',

      '!src/\*\*/\*.test.js'

    ]

  };

# Execução dos Testes de Mutação

Adição: script ao package.json:

"scripts": {

"test": "jest",

"mutation": "stryker run"

}

npm run mutation

# GitHub Repository

<https://github.com/cavalcante2021/impacta/tree/master>