

**FACULDADES INTEGRADAS DE BAURU**

CIÊNCIA DA COMPUTAÇÃO

Test Driven Development

Marcos Leandro da Cruz Barboza

Trabalho de Engenharia de software do curso de Ciência da Computação das Faculdades Integradas de Bauru – FIB.

BAURU - SP

2019

**Sumário**

**1.** **Introdução………………………………………………………………………..3**

1.1 Objetivos Gerais………………………………………………………………………….3

1.2 Motivos para o uso………………………………………………………………………….3

**2.** **Desenvolvimento………………………………………………………………...4**

2.1 Implementação……………………………………………………………………...4-5

**3.** **Tecnologias utilizadas…………………………………………………………..6**

3.1 Node.js…………………………………………………………………………………....6

3.2 Jest…………………………………………………………………………………....6

3.3 Express…………………………………………………………………………………....6

3.4 MongoDB………………………………………………………………………………...6

3.5 Babel……………………………………………………………………………………...6,

3.6 Docker…………………………………………………………………………………...6,

**4.** **Referências 7**

**1. Introdução**

1.1 Objetivos Gerais

TDD é o Desenvolvimento Orientado por Testes (Test Driven Development). Desenvolvemos o nosso software baseado em testes que são escritos antes do nosso código de produção.

TDD se baseia em pequenos ciclos de repetições, onde para cada funcionalidade do sistema um teste é criado antes. Este novo teste criado inicialmente falha, já que ainda não temos a implementação da funcionalidade em questão e, em seguida, implementamos a funcionalidade para fazer o teste passar!

1.2 Motivos para o uso

Temos diversos ganhos com esta estratégia como:

Feedback rápido sobre a nova funcionalidade e sobre as outras funcionalidades existentes no sistema

Código mais limpo, já que escrevemos códigos simples para o teste passar

Segurança no Refactoring pois podemos ver o que estamos ou não afetando

Segurança na correção de bugs

Maior produtividade já que o desenvolvedor encontra menos bugs e não desperdiça tempo com depuradores

Código da aplicação mais flexível, já que para escrever testes temos que separar em pequenos "pedaços" o nosso código, para que sejam testáveis, ou seja, nosso código estará menos acoplado.

**2. Implementação**

Vamos estar utilizando Node.js juntamente com Jest para estar realizando nossos testes.

Para instalar o jest execute o comando: **npm install jest** e logo em seguida execute o comando **jest init** para estarconfigurando o **jest.config.js** conforme as necessidades do seu projeto.

Crie uma pasta com o nome de \_\_errors\_\_ em seu projeto, logo em seguida crie um arquivo dentro desta pasta com a extensão test.js.

Dentro deste arquivo vamos executar um simples teste com o seguinte formato:

**describe**("calculadora", () => {

**it**("Soma", () => {

const x = 5;

const y = 1;

const soma = x + y;

**expect**(soma).**toBe**(6);

});

});

Esta é uma simples estrutura de testes onde o describe é a swith de testes onde se executa vários casos de teste informados pelo comando it.

Para executar o teste utilize o comando **npm test,** nosso teste irá passar com sucesso! Pois estava esperando que a soma fosse 6.

Agora vamos ver como utilizar o jest para ajudar em nossa aplicação!

Primeiro criei o seguinte teste:

**describe**("Register", () => {

**it**("User", **async** () => {

const **user** = **await** usuario.create({

nome: "marcos",

email: "leandrobarboza@gmail.com",

senha: "12345678",

dataNascimento: "24061997",

telefone: "40028922"

});

**const** response = **await** request(app)

.**post**("/cadastrar")

.**send**({

nome: user.nome,

email: user.email,

senha: user.senha,

dataNascimento: user.dataNascimento,

telefone: user.telefone

});

**expect**(response.status).**toBe**(200);

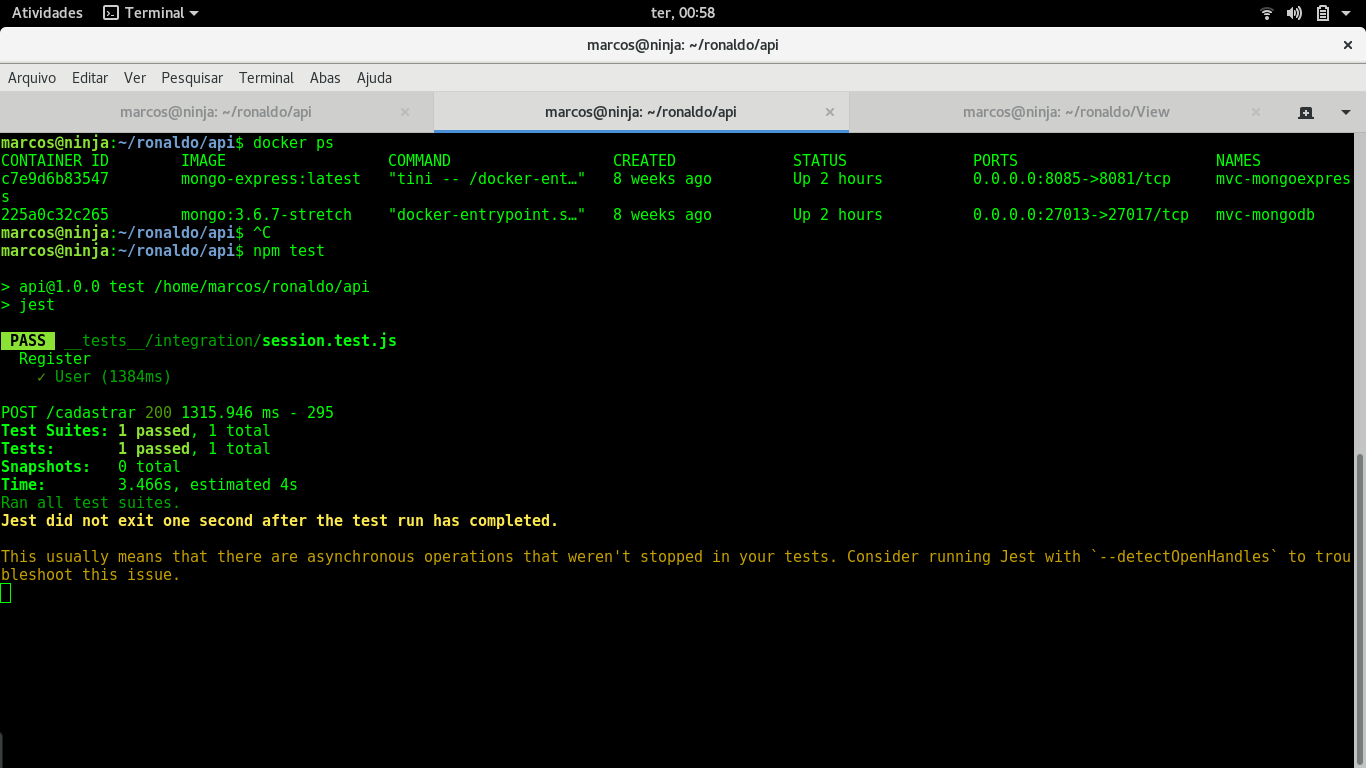
});

});

Em seguida criamos uma model de usuário com os seguintes campos:

nome, email, senha, dataNascimento, e telefone.

E uma controller que irá criar um usuário.

O teste acima garante que minha controller de usuário crie um novo usuário com os parâmetros passados em user. Logo após execute o comando npm test no console se a resposta do servidor for 200 o teste passou com a seguinte mensagem:

**3. Tecnologias utilizadas**

3.1 Node.js

Node.js é uma plataforma construída sobre o motor JavaScript do Google Chrome para facilmente construir aplicações de rede rápidas e escaláveis. Node.js usa um modelo de I/O direcionada a evento não bloqueante que o torna leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos.

3.2 Jest

Jest é um framework de testes com baterias incluídas criado pelo Facebook. ... Usando Nightmare, nós podemos rodar um navegador sem GUI (mais conhecido como headless) e realizar várias ações através de qualquer framework de testes baseado em NodeJS. Jest é um desses frameworks.

3.3 Express

Express é uma estrutura de aplicativo da web para o Node.js, lançada como software livre e de código aberto sob a licença MIT. Ele é projetado para construir aplicativos da Web e APIs. Ele foi chamado de estrutura de servidor padrão de fato para o Node.js.

3.4 MongoDB

MongoDB é um software de banco de dados orientado a documentos livre, de código aberto e multiplataforma, escrito na linguagem C++. Classificado como um programa de banco de dados NoSQL, o MongoDB usa documentos semelhantes a JSON com esquemas.

3.5 WebPack

O Webpack é um bundler de módulo JavaScript de código aberto. Seu principal objetivo é agrupar arquivos JavaScript para uso em um navegador, mas também é capaz de transformar, empacotar ou empacotar qualquer recurso ou ativo.

3.6 Babel

O Babel é um compilador JavaScript gratuito e de código aberto e transpiler configurável usado no desenvolvimento web.

3.7 Docker

Docker é uma tecnologia de software que fornece contêineres, promovido pela empresa Docker, Inc. O Docker fornece uma camada adicional de abstração e automação de virtualização de nível de sistema operacional no Windows e no Linux.

**Referências**

[**Docker**](https://www.mundodocker.com.br/o-que-e-docker/)[**Express**](https://expressjs.com/pt-br/)[**Babel**](https://desenvolvedor.expert/o-que-eh-es6-66c8d7631a0b)[**WebPack**](https://medium.com/rocketseat/entendendo-e-dominando-o-webpack-4b2e8b3e02da)[**MongoDB**](https://blog.umbler.com/br/boas-praticas-com-mongodb/)[**Node.js**](https://nodejs.org/en/)[**Jest**](https://jestjs.io/)