

tryber / sd-03-block11-project-jest

generated from [betrybe/sd-0x-project-jest](#)

☆ 0 stars 🔗 0 forks

☆ Star

🔔 Notifications

<> Code

! Issues

🔗 Pull requests 52

▶ Actions

📁 Projects

🛡 Security



🔗 master ▼

Go to file



mhamaji Merge branch 'master' of github.com:tryber/sd-03-block1...



on 17 Apr 2020

🕒 5

[View code](#)

README.md

Boas vindas ao repositório do projeto de Jest Assíncrono e Mocking!

Você já usa o GitHub diariamente para desenvolver os exercícios, certo? Agora, para desenvolver os projetos, você deverá seguir as instruções a seguir. Fique atento a cada passo, e se tiver qualquer dúvida, nos envie por *Slack!* #vqv 🚀

Aqui você vai encontrar os detalhes de como estruturar o desenvolvimento do seu projeto a partir desse repositório, utilizando uma branch específica e um *Pull Request* para colocar seus códigos.

Instruções para entregar seu projeto:

ANTES DE COMEÇAR A DESENVOLVER:

1. Clone o repositório

- `git clone https://github.com/tryber/sd-03-block11-project-jest.git` .
- Entre na pasta do repositório que você acabou de clonar:

- `cd sd-03-block11-project-jest`

2. Crie uma branch a partir da branch `master`

- Verifique que você está na branch `master`
 - Exemplo: `git branch`
- Se não estiver, mude para a branch `master`
 - Exemplo: `git checkout master`
- Agora, crie uma branch onde você vai guardar os `commits` do seu projeto
 - Você deve criar uma branch no seguinte formato: `nome-de-usuario-nome-do-projeto`
 - Exemplo: `git checkout -b joaozinho-project-jest`

4. Quando fizer mudanças, adicione-as ao `stage` do Git e faça um `commit`

- Verifique que as mudanças ainda não estão no `stage`
 - Exemplo: `git status` (devem aparecer listados os novos arquivos em vermelho)
- Adicione o novo arquivo ao `stage` do Git
 - Exemplo:
 - `git add .` (adicionando todas as mudanças - *que estavam em vermelho* - ao stage do Git)
 - `git status` (devem aparecer listados os arquivos em verde)
- Faça o `commit` inicial
 - Exemplo:
 - `git commit -m 'iniciando o projeto. VAMOS COM TUDO :rocket:'` (fazendo o primeiro commit)
 - `git status` (deve aparecer uma mensagem tipo *nothing to commit*)

5. Adicione a sua branch com o novo `commit` ao repositório remoto

- Usando o exemplo anterior: `git push -u origin joaozinho-project-jest`

6. Crie um novo `Pull Request` (PR)

- Vá até a página de *Pull Requests* do [repositório no GitHub](#)
- Clique no botão verde "New pull request"
- Clique na caixa de seleção "Compare" e escolha a sua branch **com atenção**
- Clique no botão verde "Create pull request"
- Adicione uma descrição para o *Pull Request*, um título claro que o identifique, e clique no botão verde "Create pull request"

- Não se preocupe em preencher mais nada por enquanto!
- Volte até a [página de Pull Requests do repositório](#) e confira que o seu *Pull Request* está criado

Entregáveis

Para entregar o seu projeto você deverá criar um Pull Request neste repositório. Este Pull Request deverá conter as suas modificações feitas **APENAS** sobre os arquivos da pasta `test`.

Requisitos do projeto

A seguir, estão listados todos os requisitos do projeto. Leia-os atentamente e siga à risca o que for pedido.

Nesse projeto vocês vão:

- Revisar seu conhecimento acerca de testes utilizando o Jest;
- Aplicar testes em funções assíncronas;
- 'Mockar' funções;
- 'Mockar' APIs.

Seu projeto só será avaliado se estiver passando pelos *checks* do **CodeClimate** e do **TravisCI**.

ATENÇÃO Não modifiquem os arquivos da pasta `src`, o objetivo do projeto é que vocês trabalhem apenas com a pasta `test`.

1. Jest Assíncrono

Complete os testes do arquivo `test/asyncJest.spec.js` para que funcionem com código assíncrono.

2. Mock Functions

Crie mock functions no arquivo `test/mockFunctions.spec.js` para que os testes mockados 'sobrescrevam' o código definido na pasta `src`. A idéia é que as funções criadas a partir do Jest tenham prioridade na sua execução.

3. Mock APIs

Crie um API mock no arquivo `test/mockApi.spec.js` para que os testes do Jest utilizem retornos de API fixos e independentes de requisições.



Exemplo de resposta da API randomuser.me:

```
{
  gender: 'female',
  name: { title: 'Ms', first: 'Deborah', last: 'Hanson' },
  location: {
    street: { number: 1299, name: 'Rochestown Road' },
    city: 'Birr',
    state: 'Wicklow',
    country: 'Ireland',
    postcode: 16223,
    coordinates: { latitude: '26.2451', longitude: '45.2995' },
    timezone: {
      offset: '+5:30',
      description: 'Bombay, Calcutta, Madras, New Delhi'
    }
  },
  email: 'deborah.hanson@example.com',
  login: {
    uuid: '45db2b1f-1c9a-4a80-9572-e46614f86c30',
    username: 'bluewolf366',
    password: 'iverson3',
    salt: 'XK00Gc2x',
    md5: '8cb7b4686f3869247b3ed189de780ea6',
    sha1: 'c24641f415cf36f4494ea4007fb3d77b47a6aad5',
    sha256: 'a7bdd079ead0adf21f30cee5b94e5581a9fa0d5fc8b3c1881dbc864dabc55a80'
  },
  dob: { date: '1965-10-01T06:35:49.694Z', age: 55 },
  registered: { date: '2009-02-11T05:48:39.772Z', age: 11 },
  phone: '021-953-7205',
  cell: '081-160-6277',
  id: { name: 'PPS', value: '0109675T' },
  picture: {
    large: 'https://randomuser.me/api/portraits/women/7.jpg',
    medium: 'https://randomuser.me/api/portraits/med/women/7.jpg',
    thumbnail: 'https://randomuser.me/api/portraits/thumb/women/7.jpg'
  },
  nat: 'IE'
}
```

4. Setup e Teardown

Intercale funções entre os testes definidos no arquivo `test/setupTeardown.js`.

DURANTE O DESENVOLVIMENTO



-  **PULL REQUESTS COM ISSUES NO CODE CLIMATE NÃO SERÃO AVALIADAS, ATENTE-SE PARA RESOLVÊ-LAS ANTES DE FINALIZAR O DESENVOLVIMENTO!** 
 - Faça `commits` das alterações que você fizer no código regularmente
 - Lembre-se de sempre após um (ou alguns) `commits` atualizar o repositório remoto
 - Os comandos que você utilizará com mais frequência são:
 - i. `git status` (para verificar o que está em vermelho - fora do stage - e o que está em verde - no stage)
 - ii. `git add` (para adicionar arquivos ao stage do Git)
 - iii. `git commit` (para criar um commit com os arquivos que estão no stage do Git)
 - iv. `git push -u nome-da-branch` (para enviar o commit para o repositório remoto na primeira vez que fizer o `push` de uma nova branch)
 - v. `git push` (para enviar o commit para o repositório remoto após o passo anterior)
-

DEPOIS DE TERMINAR O DESENVOLVIMENTO (OPCIONAL)

Para "**entregar**" seu projeto, siga os passos a seguir:

- Vá até a página **DO SEU Pull Request**, adicione a label de "`code-review`" e marque seus colegas
 - No menu à direita, clique no *link* "**Labels**" e escolha a *label* **code-review**
 - No menu à direita, clique no *link* "**Assignees**" e escolha **o seu usuário**
 - No menu à direita, clique no *link* "**Reviewers**" e digite `students`, selecione o time `tryber/students-sd-03`

Se ainda houver alguma dúvida sobre como entregar seu projeto, [aqui tem um video explicativo](#).

 Lembre-se que garantir que todas as *issues* comentadas pelo CodeClimate estão resolvidas! 

REVISANDO UM PULL REQUEST



À medida que você e as outras pessoas que estudam na Trybe forem entregando os projetos, vocês receberão um alerta via Slack para também fazer a revisão dos Pull Requests dos seus colegas. Fiquem atentos às mensagens do "Pull Reminders" no Slack!

Use o material que você já viu sobre [Code Review](#) para te ajudar a revisar os projetos que chegaram para você.

Releases

No releases published

Packages

No packages published

Languages

