CEFET/RJ Nova Friburgo – Bacharelado em Sistemas de Informação

Projeto Integrador de Sistemas – Prof. Thiago Delgado Pinto – 2019/1 – Trabalho Individual

Projeto Lista de Tarefas

Versão 1 de 18/03/2019

PRAZO FINAL

Versão 1: **26/03/2019** (terça-feira), via criação da *tag* 1.0.0. Após essa data, descontos devem ser aplicados.

OBJETIVO

Construir, individualmente, um pequeno projeto com uso de boas práticas de Engenharia de Software, visando dar base para a construção de um projeto maior e mais complexo.

IDEIA

Criar uma aplicação de página única (*single-page application*, SPA) que contenha uma lista de tarefas. Cada tarefa deve conter uma descrição e a indicação se ela ficou pronta. Deve ser possível cadastrar, remover e alterar a descrição de tarefas, bem como trocar a indicação de se a tarefa está pronta.

DETALHES

A lógica da aplicação e do desenho da interface de usuário (renderização no navegador) devem ficar separados. O código da lógica da aplicação *não* pode conhecer, chamar, ou ter dependências de quaisquer partes relacionadas à interface de usuário (IU). Obviamente, a IU deve utilizar o comportamento da lógica da aplicação e prover sua lógica de desenho, para dar "vida" à aplicação.

Na lógica da aplicação, o armazenamento das tarefas deve ser abstraído como uma interface, de forma que seja possível trocá-lo facilmente, bastando mudar a implementação usada para instanciá-la. A API desta interface deve trabalhar com Promessas (Promise). Na **versão 1** da aplicação, deve haver uma implementação para armazenar as tarefas no próprio navegador, usando localStorage.

PRECEITOS BÁSICOS

Os preceitos abaixo devem ser seguidos:

i. Não deve ser tomado como base nenhum projeto encontrado na web.
ii. Todo comportamento útil implementado deve ser verificado por testes automatizados – deve haver testes unitários (Jest) e testes de interface de usuário (Concordia, CodeceptJS, ou outro).
iii. Cada funcionalidade (ex.: adicionar tarefa, alterar tarefa, remover tarefa, trocar pronto) deve ter sua própria Issue, no projeto GitHub/GitLab.
iv. Cada funcionalidade deve ser implementada em um *branch* separado no controle de versões.
v. O *branch* `master` não deve ser usado como *branch* para criar funcionalidades.
vi. A cada versão estável, deve ser aumentada a versão menor (*minor*) do projeto e uma tag desta versão tem que ser criada no repositório.
vii. Todas as classes devem receber comentários no estilo JsDoc, exceto métodos *getters* e *setters*.
viii. Dúvidas devem ser discutidas pelo canal #duvidas do nosso grupo no Slack. A sua dúvida pode ser a de outros colegas.

ESTRUTURA BÁSICA

Repositório

Criar um repositório "`lista-tarefas`" no GitHub ou GitLab. Adicionar o usuário "`thiagodp`" ao projeto, com permissão mínima de *desenvolvedor*.

Início do projeto

Recomenda-se começar o projeto a partir do conteúdo do arquivo teste.zip, criado em aula. Veja o arquivo `leiame.txt` (contido em `teste.zip`), para mais instruções.

Organização sugerida

```
lista-tarefas

  ├── doc                ← documentação do projeto
  ├── src                ← código do projeto
  │    ├── __tests__     ← fica os testes
  │    │      └── e2e    ← testes e2e
  │    ├── app           ← código da aplicação (TypeScript)
  │    ├── assets        ← css e imagens
  │    └── index.html
  ├── .gitignore         ← declarar a pasta node_modules e outras geráveis
  ├── package.json
  └── ...                ← outros arquivos de configuração do projeto
```

Tecnologias básicas recomendadas

- Parcel, para empacotar a aplicação

- TypeScript, para criar a aplicação

- Jest, para testar aplicação

- Bootstrap, Material, ou uma combinação de ambos, para construir a interface

Padrões de codificação

- Estilo de codificação TypeScript proposto nesse link. Recomenda-se usar o Prettier, para garantir que o estilo está sendo seguido.

DÚVIDAS

Favor enviar dúvidas pelo grupo do Slack, via mensagem direta ou canais disponíveis (#general ou #duvidas). Eventualmente mais detalhes podem ser publicados. Procurem ser proativos e discutirem dúvidas em grupo.