Trabalho Front

React – Componentes, Roteamento, Redux e HTTP V1.1

Universidade Federal do Ceará

Projeto de Interfaces Web

Prof. Victor Farias

**Entrega** via **Moodle2**  
**Data de entrega**: 11/09/20

**Instruções**: Entregar a pasta inteira zipada excluindo apenas o diretório build e o node\_modules

**Componentes Básicos**

1. **Componente App**
   1. Componente **app** deve receber todos os componentes criados
2. **Componente Navegador superior**
   1. Logo da sua rede social à esquerda
   2. Logo após, vem 2 botões:
      1. Linha do tempo
      2. Postar
   3. Nome da pessoa logada à direita
3. **Componente Galeria de posts**
   1. Abaixo do navegador mostre os posts dos seus amigos
   2. O componente Galeria de posts usa um novo componente Post para exibir as informações de cada post
   3. Cada post deve conter três informações
      1. A pessoa que enviou
      2. A mensagem que a pessoa postou
      3. Quantidade de likes

**Props**

1. **Componente Post**
   1. Cada post deve receber como props o id da mensagem, o texto da mensagem, nomePessoa e qtdLikes e guardar em **state**.
   2. Cada post deve indicar o número de likes que as pessoas já deram nesse post. qtdLikes inicializa em 0.
   3. Cada post deve ter um botão ou ícone para like.

**Componente Galeria de Posts**

1. Esse componente deve conter como estado (atributo **state**) um lista dos dados dos posts. (obs: mais na frente, tiraremos os posts daqui)  
   state = {  
    posts: [  
    {id:1, nomePessoa:”Joao”, qtdLikes=1},  
    {id:2, nomePessoa:”Tiago”, qtdLikes=3},

{id:3, nomePessoa:”Camila”, qtdLikes=2},  
 ]  
 }

1. Esse componente deve renderizar todos os posts a partir com os dados dessa lista

**Roteamento**

1. Crie as seguintes rotas na sua aplicação:
   1. **“/”** que leva para a página linha do tempo
   2. **“/postar”** que leva para uma nova página para postar novos posts
2. Barra de navegação
   1. A barra de navegação deve conter dois botões “Linha do tempo” e “Postar” que levam para suas respectivas páginas sem atualizar a página. Deve estar incluída nas duas páginas.
   2. O botão “Linha do tempo” deve ativado quando estiver na página da linha do tempo e o botão “Post” deve estar ativado quando estiver na página Postar.
3. Página Linha do Tempo – componente Post
   1. Adicionar botão de excluir em cada Post
4. Página Postar
   1. Como conteúdo, ele contém um formulário com:
   2. Campo para inserir nome do usuário
   3. Campo para inserir mensagem
   4. Botão de submeter

**Redux**

1. Configurar Redux para a aplicação segundo slides/aula.
   1. Adicionar componente Provider em index.js
   2. Criar store global
      1. Adicionar atributo posts
      2. Inicializar posts como uma lista vazia
2. Criar ação FETCH\_POSTS
3. Criar ação FETCH\_POSTS\_SUCESSO
   1. Essa ação tem como payload uma lista de alunos
   2. O reducer referente a essa ação deve atualizar o estado global com essa nova lista de alunos
   3. Obs: somente essa ação vai ter um reducer correspondente
4. Criar ação ADICIONAR\_POST
   1. ADICIONAR\_POST tem como payload o Nome e mensagem
5. Criar ação EXCLUIR\_POST
   1. EXCLUIR\_POST tem como payload o **id** do post a ser deletado
6. Criar ação DAR\_LIKE
   1. DAR\_LIKE tem com payload o **id** do post que recebeu o like

**HTTP**

1. Ação FETCH\_POSTS
   1. Ação FETCH\_POSTS deve enviar um requisição do tipo GET para [http://rest.learncode.academy/api/seu-nome/posts /](http://rest.learncode.academy/api/seu-nome/posts%20/)
   2. Ao terminar, deve-se despachar a ação FETCH\_POSTS\_SUCESSO setando como payload a lista de alunos recebida do servidor
2. Ação ADICIONAR\_POST
   1. Ação ADICIONAR\_POST deve enviar uma requisição do tipo POST para <http://rest.learncode.academy/api/seu-nome/alunos/>
   2. Corpo deve ter formato {nome: <nome-do-usuario >, mensagem: <texto-da-mensagem>, qtdLikes: 0}
   3. Ao terminar a requisição, deve-se despachar a ação FETCH\_POSTS
3. Ação EXCLUIR\_POST
   1. Ação EXCLUIR\_POST deve enviar um requisição do tipo DELETE para [http://rest.learncode.academy/api/seu-nome/alunos/**id**](http://rest.learncode.academy/api/seu-nome/alunos/id)
   2. Ao terminar a requisição, deve-se despachar a ação FETCH\_POSTS\_ALUNOS
4. Ação DAR\_LIKE
   1. Ação DAR\_LIKE deve enviar uma requisição do tipo PUT para [http://rest.learncode.academy/api/seu-nome/alunos/**id**](http://rest.learncode.academy/api/seu-nome/alunos/id)
   2. Essa requisição deve atualizar o post mantendo o id, nome do usuário e mensagem e incrementando o qtdLikes em 1.
   3. Deve-se despachar a ação FETCH\_POSTS ao terminar a requisição