Fórum de Comentários

`Comentarios.js`

- `React` e `useState` são importados do pacote 'react' para usar hooks de estado.
- `useEffect` é importado do pacote 'react' para lidar com efeitos colaterais em componentes funcionais.
- `axios` é importado para fazer solicitações HTTP.
- `Formulario` e `FormularioAtualizacao` são importados de arquivos locais.
- './App.css' é importado para importar um arquivo de estilo CSS.

- Componente Comentários.js
  - O componente é definido como uma função React.
  - Usa hooks de estado para definir os estados iniciais dos comentários, comentário para atualização, página atual e indicador de carregamento.
  - O hook `useEffect` é usado para chamar a função `fetchComments` quando o componente é montado pela primeira vez.
- Função fetchComments
  - Faz uma solicitação GET para a URL 'https://jsonplaceholder.typicode.com/comments' usando o Axios.
  - Se a solicitação for bem-sucedida, os dados de resposta são definidos como o estado dos comentários.
  - Se ocorrer algum erro, o erro é registrado no console.
- Funções de manipulação de comentários
  - `handleCommentAdded` é chamado quando um novo comentário é adicionado. Adiciona o novo comentário ao estado dos comentários.
  - `handleUpdateComment` é chamado ao atualizar um comentário existente. Faz uma solicitação PUT para atualizar o comentário na API e atualiza o estado dos comentários com o comentário atualizado.
  - `handleDeleteComment` é chamado ao excluir um comentário existente. Faz uma solicitação DELETE para remover o comentário da API e atualiza o estado dos comentários removendo o comentário correspondente.
  - `handleEditComment` é chamado ao editar um comentário. Define o comentário selecionado como o estado do comentário para atualização.
  - `handleCancelEdit` é chamado para cancelar a edição do comentário. Define o estado do comentário para atualização como nulo.

- Função fetchMoreComments:
    - É chamado quando o usuário rola até o final da página.
    - Faz uma solicitação GET para a próxima página de comentários usando a variável de estado `page`.
    - Os comentários da resposta são adicionados aos comentários existentes e a página é atualizada.
    - O indicador de carregamento é definido como verdadeiro durante a solicitação e falso após a conclusão.
- Hook useEffect para detecção de rolagem:
    - É usado para adicionar um ouvinte de evento de rolagem à janela quando o componente é montado.
    - Quando o usuário rola até o final da página, a função `fetchMoreComments` é chamada.
- Renderização JSX:
    - Renderiza um formulário (`Formulario`) para adicionar novos comentários.
    - Renderiza a lista de comentários com base no estado dos comentários.
    - Os comentários são mapeados e renderizados em elementos `<li>` com informações sobre o nome, email e mensagem do comentário.
    - Se um comentário estiver em modo de edição, o componente `FormularioAtualizacao` é renderizado para atualizar o comentário.
    - Caso contrário, são renderizados botões para editar e excluir o comentário.
    - Se o estado de carregamento for verdadeiro, uma mensagem "Carregando..." é exibida.
    - Se houver um comentário em modo de edição, são exibidos detalhes do comentário e um botão "Cancelar".

`FormularioAtualizacao.js`

    - O componente é definido como uma função React.
    - Recebe duas propriedades: `comment` (comentário a ser atualizado) e `onUpdateComment` (função para atualizar o comentário).
    - Usa o hook de estado `useState` para definir os estados iniciais do corpo do comentário, mensagem de sucesso e mensagem de erro.
- Função handleSubmit:
    - É chamada quando o formulário de atualização é enviado.
    - Previne o comportamento padrão do formulário.
    - Cria um objeto `updatedComment` com base no comentário existente e o novo corpo fornecido.

    - Chama a função `onUpdateComment` passando o ID do comentário e o objeto `updatedComment` como argumentos.
    - Se a atualização for bem-sucedida, define a mensagem de sucesso.
    - Se ocorrer algum erro, define a mensagem de erro e registra o erro no console.
- Renderização JSX:
    - Renderiza um formulário com um campo de entrada de texto para o corpo do comentário.
    - O valor do campo de entrada é definido como o estado `body` e é atualizado usando o evento `onChange` para chamar a função `setBody`.
    - Quando o formulário é enviado, a função `handleSubmit` é chamada.
    - Renderiza um botão "Atualizar" para enviar o formulário.
    - Se houver uma mensagem de sucesso, ela é renderizada como um parágrafo.
    - Se houver uma mensagem de erro, ela é renderizada como um parágrafo.
- Exportação:
    - Exporta o componente `FormularioAtualizacao` como padrão.

Esse componente é usado para renderizar um formulário de atualização de comentários, permitindo que o usuário atualize o corpo do comentário. Quando o formulário é enviado, a função onUpdateComment é chamada para atualizar o comentário. A mensagem de sucesso ou erro é exibida com base no resultado da atualização do comentário.

`Formulario.js`

    - O componente é definido como uma função React.
    - Recebe uma propriedade chamada `onCommentAdded`, que é uma função para lidar com a adição de comentários.
- Hooks de estado:
    - São usados vários hooks de estado para gerenciar os valores dos campos do formulário, mensagens de sucesso e erro.
    - `name` representa o valor do campo "Nome".
    - `email` representa o valor do campo "Email".
    - `message` representa o valor do campo "Mensagem".
    - `successMessage` representa a mensagem de sucesso exibida após o envio do formulário.
    - `errorMessage` representa a mensagem de erro exibida quando há campos vazios ou ocorre um erro durante o envio do formulário.
- Função handleSubmit:

    - É chamada quando o formulário é enviado.
    - Previne o comportamento padrão do formulário.
    - Valida os campos do formulário verificando se `name`, `email` e `message` têm valores preenchidos. Se algum campo estiver vazio, define a mensagem de erro correspondente e retorna.
    - Cria um objeto `user` com base nos valores dos campos preenchidos.
    - Faz uma solicitação POST para a API usando `axios`, enviando o objeto `user` como dados.
    - Se a solicitação for bem-sucedida, define a mensagem de sucesso, limpa os campos do formulário, chama a função `onCommentAdded` com os dados de resposta e limpa a mensagem de erro.
    - Se ocorrer um erro na solicitação, define a mensagem de erro e registra o erro no console.
- Renderização JSX:
    - Renderiza um formulário com campos de entrada para "Nome", "Email" e "Mensagem".
    - Os valores dos campos são definidos com base nos estados correspondentes e são atualizados usando o evento `onChange` para chamar as funções de atualização de estado.
    - Quando o formulário é enviado, a função `handleSubmit` é chamada.
    - Renderiza um botão "Enviar" para enviar o formulário.
    - Se houver uma mensagem de sucesso, ela é renderizada como um parágrafo com a classe "success-message".
    - Se houver uma mensagem de erro, ela é renderizada como um parágrafo com a classe "error-message".
- Exportação:
    - Exporta o componente `Formulario` como padrão.

Esse componente permite que o usuário preencha um formulário de envio de comentários. Os campos do formulário são validados antes do envio e as mensagens de sucesso ou erro são exibidas com base no resultado da solicitação. O componente utiliza o pacote axios para fazer chamadas à API e a função onCommentAdded é chamada para lidar com a adição de comentários.