TodoList Frontend React - Roteiro 04 – Aula 03 - Continuação

Instalando as dependências necessárias

Crie na pasta componentes, um arquivo chamado TodoForm.jsx com o conteúdo abaixo:

```
npm install @fortawesome/react-fontawesome @fortawesome/free-solid-svg-icons
```

Incluindo o ícone de Lixeira nas tarefas

Edite o arquivo chamado TodoList.jsx com o conteúdo abaixo:

```
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome'
import { faTrash } from '@fortawesome/free-solid-svg-icons'
import React from 'react'

export const TodoList = ({task}) => {
  return (
    <div className="Todo">
        <p className={`${task.completed ? "completed" : "incompleted"}`}
>{task.description}</p>
        <div>
          <FontAwesomeIcon className="delete-icon" icon={faTrash}/>
        </div>
    </div>
  )
}
```

Visualizando a alteração

Repare que agora temos a ação para excluir uma tarefa, mas ela ainda não esta funcional, conforme abaixo:

Implementando a exclusão

Edite o arquivo TodoWrapper .jsx, incluindo os seguintes trechos de código.

1. O código abaixo é responsável por excluir um item especifico, da lista chamada **todos**, que contém todos os itens da lista

```
const deleteTodo = (id) => setTodos(todos.filter((todo) => todo.id !== id));
```

2. Inclua o função deleteTodo no item TodoList, conforme código abaixo:

```
          <TodoList
             key={todo.id}
             task={todo}
             deleteTodo={deleteTodo}
           />
```

O arquivo TodoWrapper.jsx ficará assim:

```
import React, { useState }  from 'react';
import { TodoForm } from './TodoForm';
import { TodoList } from './TodoList';
import { v4 as uuidv4 } from "uuid";

export const TodoWrapper = () => {
  const [todos, setTodos] = useState([]);

  const addTodo = (todo) => {
    setTodos([
      ...todos,
      { id: uuidv4(), description: todo, completed: false },
    ]);
  }

  const deleteTodo = (id) => setTodos(todos.filter((todo) => todo.id !== id));

  return (
    <div className='TodoWrapper'>
      <h1>Lista de Tarefas</h1>
      <TodoForm addTodo={addTodo}/>
      {todos.map((todo) =>
          <TodoList
            key={todo.id}
            task={todo}
            deleteTodo={deleteTodo}
          />
      )
      }
    </div>
  );
}
```

Não se esqueça de validar no navegador, se sua aplicação continua funcionando. Ainda não teremos nenhuma alteração de comportamento (botão excluir ainda não vai funcionar), mas é importante ter certeza que não quebrou nada.

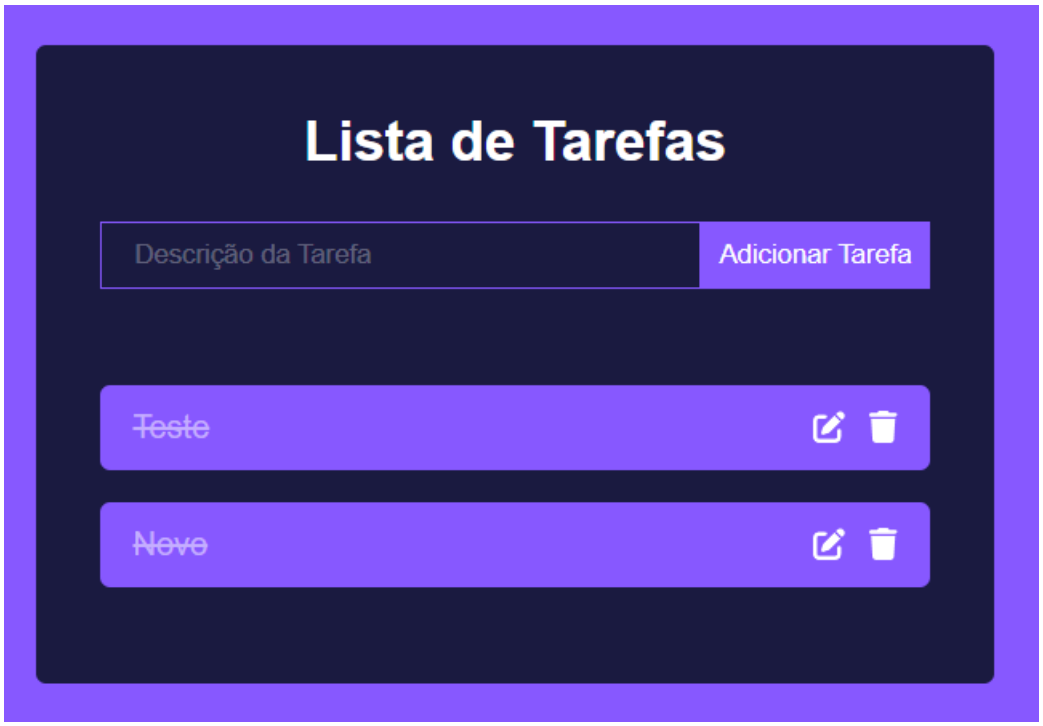
Edite novamente o arquivo TodoList.jsx, agora preparando ele para receber o todo deleteTodo e implementando sua chamada no click do ícone.

Implementando a conclusão da tarefa

A função para edição deverá ser conforme o código abaixo.

```
const toggleComplete = id => {
    const newTodos = todos.map(todo => todo.id === id ? {...todo, completed:
!todo.completed} : todo);
    setTodos(newTodos);
    localStorage.setItem('todos', JSON.stringify(newTodos));
}
```

Agora basta seguir os mesmos passos que foram feitos na exclusão da tarefa, que teremos a funcionalidade funcionando corretamente, conforme a imagem abaixo:



Incluindo o ícone de Edição nas tarefas

Edite o arquivo chamado TodoList.jsx e inclua agora o ícone de edição, repetindo os mesmos passos do item de lixeira, porém com o ícone abaixo. Não se esqueça do import.

```
          <FontAwesomeIcon className="edit-icon" icon={faPenToSquare} />
```

A tela deverá ficar conforme abaixo:

Implementando a edição da tarefa

A função para edição deverá ser conforme o código abaixo e basta seguir os mesmos passos que foram realizados na exclusão.

```
  const editTodo = (id) => {
    setTodos(
      todos.map((todo) =>
        todo.id === id ? { ...todo, isEditing: !todo.isEditing } : todo
      )
    );
  }
```

Ao final, seu código TodoWrapper deverá esta conforme abaixo:

```
import React, { useState }  from 'react';
import { TodoForm } from './TodoForm';
import { TodoList } from './TodoList';
import { v4 as uuidv4 } from "uuid";

export const TodoWrapper = () => {
  const [todos, setTodos] = useState([]);

  const addTodo = (todo) => {
```

```
    setTodos([
      ...todos,
      { id: uuidv4(), description: todo, completed: false },
    ]);
  }

  const deleteTodo = (id) => setTodos(todos.filter((todo) => todo.id !== id));

  const editTodo = (id) => {
    setTodos(
      todos.map((todo) =>
        todo.id === id ? { ...todo, isEditing: !todo.isEditing } : todo
      )
    );
  }

  return (
    <div className='TodoWrapper'>
      <h1>Lista de Tarefas</h1>
      <TodoForm addTodo={addTodo}/>
      {todos.map((todo) =>
          <TodoList
            key={todo.id}
            task={todo}
            deleteTodo={deleteTodo}
            editTodo={editTodo}
          />
      )
      }
    </div>
  );
}
```

Se o seu código de edição, ainda não funciona, é normal. Pois apenas implementamos uma forma de uma tarefa chamar a função de edição. Mas ainda não implementamos um componente de edição, nem uma forma da função de edição, acionar o componente de edição dos dados, mas não vamos preocupar com isso agora