| Uma imagem contendo placar, desenho, relógio  Descrição gerada automaticamente | **ATIVIDADE ADICIONAL PRÁTICA – REACT** |
| --- | --- |
| REACT-2 |

**Instruções gerais:**

| 1. Utilize o Eclipse ou o STS para desenvolver os algoritmos. |
| --- |
| **Mantenha as entregas das Atividades em dia na Plataforma Canvas** |

**EXERCÍCIOS**

**Boas práticas:**

1. Leia o enunciado do exercício com atenção
2. Observe as indicações de Entrada e Saída esperadas em cada exercício
3. Observe com atenção os desenhos e diagramas inseridos nos exercícios para facilitar a compreensão
4. Utilize o Cookbook, os Vídeos da Plataforma e os Códigos guia como referências para a resolução dos exercícios
5. Caso ainda fique alguma dúvida, consulte os instrutores da sua turma pelo Discord

**Lista de exercícios**

**Exercícios sobre React props e hooks**

**Contador com Hooks**: Crie um componente funcional que exibe um contador. Utilize o hook useState para gerenciar o estado do contador e forneça botões para incrementar e decrementar o valor.

**Formulário Controlado**: Desenvolva um componente de formulário que use useState para gerenciar o estado dos campos de entrada (nome e email). Adicione validação simples para garantir que ambos os campos sejam preenchidos antes de permitir o envio do formulário.

**Componente de Listagem**: Crie um componente que receba uma lista de itens como props e exiba esses itens em uma lista ordenada. O componente deve ter uma prop title para exibir um título acima da lista.

**Temporizador com useEffect**: Implemente um temporizador que atualiza o tempo decorrido a cada segundo. Utilize o hook useEffect para configurar e limpar o intervalo de atualização.

**Controle de Tarefa com Estado**: Crie um componente que exibe uma lista de tarefas. Cada tarefa deve ter um botão para marcar a tarefa como concluída. Use useState para gerenciar o estado de conclusão de cada tarefa.

**Componente de Modo Claro/Escuro**: Desenvolva um componente que permita alternar entre os modos claro e escuro. Use o hook useState para gerenciar o tema e altere a classe do componente com base no estado.

**Uso de useMemo para Otimização**: Crie um componente que recebe uma lista de números e calcula a soma desses números. Utilize useMemo para otimizar o cálculo da soma e evitar reprocessamentos desnecessários.

**Hook Personalizado de Contador**: Implemente um hook personalizado que gerencia um contador. O hook deve fornecer funções para incrementar, decrementar e resetar o contador. Use esse hook em um componente funcional.

**Exibição Condicional com Props**: Desenvolva um componente que recebe uma prop isLoggedIn. Se isLoggedIn for verdadeiro, exiba uma mensagem de boas-vindas, caso contrário, exiba uma mensagem pedindo para fazer login.

**Formulário com Múltiplos Campos**: Crie um componente de formulário que tenha vários campos (nome, email, senha). Utilize useState para gerenciar o estado de cada campo e useEffect para validar os campos quando o formulário for enviado.