DESENVOLVIMENTO DE SOFTWARE PARA WEB - AP1

Obs.: Para cada questão de implementação, use pelo menos 3 comentários para explicar o que está ocorrendo no código. Código não comentado não será corrigido! Evite comentários óbvios. Utilize o App.js para importar e testar suas questões, assim como fizemos em sala de aula. Obedeça os nomes dos arquivos para cada questão.

Questão 01 - Em um **mesmo** arquivo chamado **Questao01.js**, crie dois componentes JSX:

- O componente **Questao01A** (crie usando função arrow)
- O componente **Questão01B** (crie usando função clássica, ou seja, function)

O componente **Questao01A** deve chamar, internamente em seu JSX, o componente **Questao01B**. Além disso, o componente **Questao01A** irá passar, via props, para o componente **Questao01B**, o seguinte objeto:

```
const lista = [
        {a:10, b:3, c: 7},
        {a:5, b:-3, c: 9},
        {a:1, b:9, c: 40}
    ]
```

Em **Questao01B** você deve mostrar, em seu JSX, o maior elemento de cada objeto JSON passado em **lista**. Mostre os maiores números em sua página, da forma que desejar.

Questão 02 - Crie um arquivo chamado **Questao02.jsx** que deverá, em seu JSX, renderizar a imagem de um Pokemon de sua preferência. Por exemplo, para renderizar o Pikachu, use o caminho:

https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/25.png

Este caminho irá mostrar o Pikachu **de frente**.

Insira um botão, em **Questao02.jsx**, que ao ser pressionado, deverá mudar a imagem frontal para imagem de costas. Eis o caminho da imagem do Pikachu de costas:

https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/back/25.png

Ao pressionar o botão novamente, a imagem volta a ficar de frente e assim por diante, alternando entre frente e costas.

Questão 03 - Seja o seguinte código a ser executado no console do navegador:

```
fetch("https://restcountries.com/v3.1/region/europe?fields=capital,population")
.then(
   (response) => {
```

```
      return response.json()
    }
)
.then(
    (data) => {
       data.map(console.log(data))
    }
)
.catch(error => console.log(error))
```

Ele retorna, em data, o seguinte objeto (o qual nesse caso é impresso em **console**):

```
[
{"capital":["Dublin"],"population":4994724},
{"capital":["Nicosia"],"population":1207361},
        ...
{"capital":["Madrid"],"population":47351567}
]
```

Ou seja, um vetor com vários objetos contendo duas propriedades: **capital**, do tipo vetor de string e **population**, do tipo numérico.

Crie um componente chamado **Questao03.jsx** que imprima na tela as duas capitais com maior **E** menor população (**population**). Use **async-await** para acessar o serviço..

Questão 04 - Faça uma cópia da **Questão03** (renomeando para **Questao04.jsx**) mas agora você não vai mais acessar um serviço externo. Você deverá criar uma **Promise,** em **Questao04.jsx,** cujo **resolve** deve retornar o seguinte Array:

```
[
{"capital":["Dublin"],"population":4994724},
{"capital":["Nicosia"],"population":1207361},
{"capital":["Madrid"],"population":47351567}
]
```

Então, no **useEffect** de **Questao04**, você irá usar a **Promise** criada com then-catch **ou** async-await. O resto do exercício será o mesmo (mostrar a capital de maior e menor população). Aproveite o seu código da questão 03!

Questão 05 - Crie um arquivo **Questao05.txt** e (salve no mesmo nível dos outros): Explique como o uso de Contextos pode resolver o problema de **PROPS DRILLING**. Use um exemplo em (pode codificar ou simplesmente desenhar e explicar. Não precisa executar)

Preencha a tabela abaixo com a quantidade de pontos que você acha que consegiu:

Questão 01 (2pts)	Questão 02 (2pts)	Questão 03 (2pts)	Questão 04 (2pts)	Questão 05 (2pts)