

## PRIMEIRA AVALIAÇÃO PARCIAL - A

### DESENVOLVIMENTO DE SOFTWARE PARA WEB PROJETO DE INTERFACES WEB

2023.1 - PROF. JEFFERSON DE CARVALHO SILVA

UFC - QUIXADÁ

Questão 01 (2.5pts)	Questão 02 (2.5pts)	Questão 03 (2.5pts)	Questão 04 (2.5pts)

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.**

**Questão 01** - Em um mesmo arquivo chamado **Questao01.jsx**, 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 alunos = [  
  { nome: "Sicrano", notas: {ap1: 8.4, ap2: 5.4} },  
  { nome: "Beltrano", notas: {ap1: 6.7, ap2: 3.5} },  
  { nome: "Fulano", notas: {ap1: 7.3, ap2: 9.2} }  
]
```

O componente **Questão01B**, ao receber o objeto **alunos**, deverá calcular a média de CADA aluno:  $(ap1 + ap2)/2$  dentro do seu hook **useEffect**. As médias serão armazenadas em um vetor de três posições que deve ser retornado ao componente **Questao01A** (comunicação filho → pai).

De posse do vetor de médias, o componente **Questao01A** deve imprimir, em seu JSX, **apenas** os nomes dos alunos com **média superior ou igual a 6.0** (use a forma que achar melhor, map ou laço comum).

Dicas:

- Em **Questao01A**, cria uma função para ser repassada a **Questao01B** a qual inicializa uma variável de estado em **Questao01A** chamada de “medias”, com a médias calculadas por **Questao01B** (comunicação filho → pai).
- Em **Questao01A**, crie uma função que retorne um JSX para renderizar todos os alunos acima da média (essa função deve fazer um **map** ou **for** e gerar um vetor de elementos JSX).
- Em **Questao01A**, crie um variável de estado “loading” para forçar o carregamento da página ao receber e vetor de médias de **Questao01B**.

**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 seu caminho 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.

Dica:

- Crie uma variável de estado “virar”, booleana, para ficar alternando entre true e false. No entanto, o mais adequado seria o useRef (opcional).

**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) => {
      console.table(data.map(
        (country) => country.capital[0] + "-" + country.population
      ))
    }
  )
  .catch(error => console.log(error))
```

Ele retorna a saída, impressa em console.table:

```
[
  { "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 capitais do países com maior **E** menor população (**population**). Use then-catch **ou** async-await.

Dica:

- Na função **callback** do último **then**, faça um laço que percorre todo o vetor **data** de objetos. Crie as variáveis numéricas locais (**let**) para armazenar a maior e menor **population** e também os seus índices do vetor **data**. Ao final do laço, de posse desses índices, acessando o vetor **data**, alimente duas variáveis de estado que representam os nomes das capitais de maior e menor população. Imprima essas variáveis em tela.

### **OBSERVAÇÃO - NO CASO DA INTERNET DO CAMPUS ESTAR OFFLINE:**

Caso a internet não esteja disponível, você deverá então criar uma **Promise**, em **Questao03**, cujo **resolve** deve retornar o seguinte Array:

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

Então, no **useEffect** de **Questao03**, 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).

**Questão 04** - Crie um arquivo **Questao04.txt** e:

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)

## ATENÇÃO

Crie um último arquivo chamado PONTOS.TXT. Neste arquivo diga quantos pontos você acha que deve ganhar em cada questão, por exemplo:

1 - 2,0

2 - 0,0

3 - 1,5

4 - 2,5

Pense da seguinte forma: cada funcionalidade não implementada (ou não respondida), retira 0,5 pts da questão.