Atividade de Participação 4 - Programação assíncrona com Javascript e Promises

> Prof. Luiz Gustavo D. de O. Véras
> Desenvolvimento de Sistemas Web (DSWI6)
> Tecnologia em Análise e Desenvolvimento de Sistemas

Atividade 1. Crie uma função que após 5 segundos dobre o resultado de um número passado como argumento. Essa função deverá retornar uma Promise. Use o *setTimeOut* como temporizador.

```
function dobrarEm5Segundos(x) {
  return new Promise( => {
    setTimeout(() => {
    });
  });
}
```

Atividade 2. Implemente uma função chamada `doAction` que retona uma Promise. A promise deve ser resolvida 2 segundos após a sua chamada e deve retornar a mensagem `olá mundo`.

Atividade 3. Crie uma função que retorne uma Promise seguindo as seguintes orientações:

- Se o argumento da função não for um número, retorne uma promessa rejeitada instantaneamente e forneça uma mensagem de "erro" aos dados (em uma string);
- Se os dados forem um número ímpar, retorne uma promessa resolvida 1 segundo depois e forneça os dados "ímpares" (em uma string);
- Se os dados forem um número par, retorne uma promessa rejeitada 2 segundos depois e forneça os dados "par" (em uma string).

Atividade 4. Considere o seguinte código

```
    firstPromise(numInt)
        .then(data => secondPromise(data))
        .then(data =>{
            console.log(data)
    }).catch(e => {
        console.log(e)
    })
```

Implemente as funções acima que retornam Promises seguindo a seguinte lógica:

- **firstPromise:** deve ser resolvido se numInt é maior que 2 e rejeitada caso contrário;
- **secondPromise:** deve ser resolvida se data + 1 é par e rejeitada caso contrário.

Atividade 5. Considere a função a seguir. Ela é uma função de soma que precisará ter como resultado um número par. Para isso, você precisará criar os seguintes callbacks:

- callbackSucesso() = trará uma mensagem de sucesso, dizendo que a operação foi concluída com sucesso e que o número somado é par.
- callBackError() = trará uma mensagem de erro, dizendo que a operação falhou, trazendo um número impar (resultado da soma).

Tome como base o seguinte código:

```
function somar(){
    if (){
        callbackSucesso();
    } else {
        callbackError();
    }
}

function callbackSucesso(){
    console.log("Resposta gerada com sucesso");
}

function callbackError(){
    console.log("");
}
```

Crie uma função que produza uma Promise para representar essa operação de somar.

> Dica: Utilize uma promise com callback que recebe como parâmetros os objetos (resolve, reject).

Atividade 6. Refatore o seguinte trecho de código para utilizar `async/await`:

```
    // Função delay aciona o .then após 1s
    const delay = () => new Promise(resolve => setTimeout(resolve, 1000));

    function umPorSegundo() {
        delay().then(() => {
            console.log('1s');
            delay().then(() => {
                console.log('2s');
                delay().then(() => {
                    console.log('3s');
                });
            })
        });
    }

    umPorSegundo();
```