# Requisitos:

https://www.figma.com/file/IIpJ82obcQ4LTpLYZO5KFllT/Jobify-Fullstack-Lab?node-id=0%3A1

# Aulas extras:

Link para assistir as aulas extras:

<https://vimeo.com/album/5882467>

Senha: devpleno

# Exercícios extras (Aula 01):

Os exercícios aqui propostos visam reforçar a prática da linguagem javascript de acordo com o que foi visto na primeira aula extra.

1. Dado um vetor de números, como poderia ser realizada a soma de todos os valores utilizando reduce.
2. Dado um vetor de números, como poderia ser realizada a soma de todos os valores pares utilizando reduce e filter.
3. Dado um vetor de números, como poderia ser realizada a soma de todos os valores ímpares utilizando reduce e filter.
4. Dado um vetor de valores, retorne um objeto com quantas vezes cada valor está presente no vetor (dica: utilize reduce)
5. Dado um vetor de valores, retorne um vetor com somente os valores únicos do vetor (aqueles que ocorrem apenas 1 vez dentro do vetor) (Dica 1: utilize reduce, filter e keys, Dica 2: escreva console.log(objeto.keys()) e veja como ele poderá te ajudar neste exercício)
6. Dado um vetor com números, retorne somente os números pares;
7. Dado um vetor com números, retorne somente os números ímpares;
8. Uma função é chamada da seguinte forma:

calculadora(10, '+', 20)

crie o corpo da função de forma que ela realize as 4 operações aritméticas

1. Modifique a calculadora do exercício anterior para que ela receba 2 números e uma função, e realize o cálculo. Exemplo:

const soma = (num1, num2) => num1+num2

const calculadoraFn = (....) => ….

calculadoraFn(10, soma, 20)

# Exercícios extra (Aula 02):

1. Em linguagens síncronas é comum a função sleep(tempo) assim, pode-se por exemplo escrever algo no console, pausar o script por x tempo e escrever algo mais algo como:

console.log('Olá!')

sleep(2000)

console.log('Depois de 2segs')

simule o sleep de maneira correta em Javascript (sem travar a thread principal) utilizando promises e uma função async/await

2) A função readdir disponível no módulo fs do Node, permite ler um diretório, e retornar seu conteúdo como um vetor. Crie uma versão em Promise da função.

Dica: a assinatura da função é: fs.readdir(caminho, (err, files) => {})

3) A lista de arquivos retornado no exercício anterior, também retorna diretórios. Crie uma função que retorne todos os sub-diretórios em um vetor (deve-se ignorar os arquivos retornados pelo readdir).

Algumas funções do fs e de Promises que te ajudarão:

* fs.stat(caminhoArquivoDiretorio, (err, stats) => {})
* stats que é retornado no callback possui um isDirectory() e um isFile()
  + (sou bonzinho né, cantei a pedra aqui ;) )
* Promise.all(vetorDePromises): aguarda um vetor de promises ser executado, e retorna uma promise que retorna para um vetor com o resultado.
* Ex:

Promise.all([promise1, promise2]).then( results => {

results[0] // resultado de promise1

results[1] // resultado de promise2

})

* este exercício é um pouco mais desafiador; é normal ter um pouco de dificuldade

3) A lista de arquivos retornado no exercício 2, também retorna diretórios. Crie uma função que retorne todos os arquivos em um vetor (deve-se ignorar os diretório retornados pelo readdir).

Algumas funções do fs e de Promises que te ajudarão:

* fs.stat(caminhoArquivoDiretorio, (err, stats) => {})
* stats que é retornado no callback possui um isDirectory() e um isFile()
  + (sou bonzinho né, cantei a pedra aqui ;) )
* Promise.all(vetorDePromises): aguarda um vetor de promises ser executado, e retorna uma promise que retorna para um vetor com o resultado.
* Ex:

Promise.all([promise1, promise2]).then( results => {

results[0] // resultado de promise1

results[1] // resultado de promise2

})

* este exercício é um pouco mais desafiador; é normal ter um pouco de dificuldade