Processo Seletivo - Rocky

Estágio em TI: Web Development

Thiago Savi Ribeiro

Agosto/2020

Explicando cada Funcionalidade

1. Lendo um Arquivo JSON com JavaScript:

Para ler o arquivo **"broken-database.json"** foi usado a função **require()**, onde foi importada para a constante **data**, que é utilizada para fazer todo o tratamento do banco de dados corrompido.

```
7
8   // Constante que recebe os dados do broken-database.json
9   const data = require('./broken-database.json')
10
```

*Figura 1: Constante **data** recebendo o arquivo "broken-database.json".*

2. Corrigindo o banco de dados:

Foram criadas as constantes **correctName**, **correctPrice** e **correctQuantity** para o tratamento dos problemas encontrados.

```
11  // Constantes que vão ajustar o banco de dados
12  const correctName = []
13  const correctPrice = []
14  const correctQuantity = []
15
```

Figura 2: Constantes para correção do banco de dados.

2.1. Tratando nomes (correctName):

Para corrigir os nomes dos produtos dentro do banco de dados foi utilizado a estrutura de repetição **for()** para varrer a constante **data** e com a função **push()** empurrar toda a **data** para a constante **correctName** onde foi utilizada a função **replace()** e o método **RegEX**, que localiza globalmente todos os dígitos errado para substituí-lo pela letra correta.

```
15
16  // função para ajuste dos Nomes dos itens
17  for (item of data){
18      correctName.push({ ...item, name: item.name
19          .replace(/æ/g, 'a')
20          .replace(/ß/g, 'b')
21          .replace(/¢/g, 'c')
22          .replace(/ø/g, 'o')
23      })
24  }
25
```

*Figura 3: Estrutura **for()** que corrige os nomes.*

2.2. Tratando preços (correctPrice):

Para corrigir os preços dos produtos dentro do banco de dados também foi utilizada a estrutura **for()** para varrer agora a constante **correctName,** onde já havia os dados com os nomes corridos, com isso usando a função **push()** foi empurrado todos os itens do banco de dados, porém agora transformando todos os itens **price** em números utilizando o objeto **Number().**

```
26  // função para passar todos os itens PRICE de String para Number
27  for (item of correctName){
28      correctPrice.push({ ...item, price: Number(item.price)
29      })
30  }
```

*Figura 4: Estrutura **for()** que corrige os preços.*

2.3. Tratando quantidades (correctQuantity):

Para inserir o item **quantity** com o valor 0 (zero) aos produtos que não havia tal item, a constante **correctPrice** que agora contém todo o banco de dados com nomes e preços corridos foi varrida pela função **for()** e empurrada para a constante **correctQuantity,** utilizando uma **estrutura condicional (if() / else())** para testar se o item **quantity** já existe e caso não exista ele é adicionado com o valor 0 (zero).

```
32  //função que adiciona o item QUANTITY com valor 0
33  for (item of correctPrice){
34      correctQuantity.push({ ...item, quantity: item.quantity? item.quantity : 0
35      })
36  }
```

*Figura 5: Estrutura **for()** que corrige as quantidades.*

2.4. Constante finalData:

Foi criada a constante **finalData** para consolidar todo o tratamento realizado para o ajuste do banco de dados corrompido.

```
38  // Constante que consolida todo o tratamento do banco de dados corrompido
39  const finalData = correctQuantity
40
```

*Figura 6: Constante **finalData** recebendo o conteúdo da constante **correctQuantity**.*

2.5. Exportar o banco de dados corrigido para um arquivo JSON (saida.json):

Para escrever um arquivo JSON com o banco de dados corrigido foi utilizada a função **fs.witterFile()** e dentro desta, a função **JSON.stringify()**, com isso é criado e convertido o arquivo **"saída.json"**.

```
41  // Função que converte para .JSON e salva o conteúdo da variável finalData em um aquivo "saida.json"
42  const fs = require('fs')
43  fs.writeFile('saida.json', JSON.stringify(finalData, null, '\t'), function(err, result) {
44    if(err) console.log('error', err)
45    else console.log("Corrected Data!")
46  })
```

Figura 7: Criação do arquivo JSON com o banco de dados corrigido.

3. Validação.

3.1. Ordenando o banco de dados:

Para colocar o banco de dados em ordem alfabética de **category** e depois por **ID,** no caso de haver mais de um produto da mesma categoria, foi utilizado o método **sort(),** que tem a função de classificar os elementos de um **array**. Dentro desta foi utilizado estruturas condicionais **if** para classificação de acordo com o valor do **return**, ou **(||),** em caso de mais de um produto ordenar os mesmos por **id**. Ao fim, no **console.log()**, foi utilizada a função **JSON.stringify()** para converter a saída para **JSON** e imprimir no terminal os dados ordenado.

```
52  finalData.sort((obj1, obj2) => {
53      if(obj1.category > obj2.category) return 1
54      if(obj1.category < obj2.category) return -1
55      || (obj1.id - obj2.id)
56      return 0
57
58  })
59  console.log(JSON.stringify(finalData, null, '\t'))
```

Figura 8: Função que ordena o banco de dados por categoria e depois por ID.

3.2. Imprimindo o valor acumulado por categoria:

Para esta função foram criadas duas constantes: **name** e **value.**

```
63  const name = []
64  const value = []
```

Figura 9: Constantes para a consolidação da Category e price.

Para varrer a constante **finalData** foi usado a estrutura de repetição **for()**, nessa foi adicionada uma estrutura condicional **if / else** que primeiro testa se não há o item **category** na constante **name,** caso essa condição seja verdadeira, a função **push()** **empurra** a categoria na constante **name,** isso também ocorre com a constante **value,**

porém nesta a função **push()** empurra a expressão "**quantity * price**", caso a condição for falsa a função lastIndexOf() adiciona o valor à constante **value**, utilizando a mesma expressão "**quantity * price**".

```
66  for(i in finalData){
67      if(!name.includes(finalData[i].category)){
68          name.push(finalData[i].category) &&
69          value.push(finalData[i].quantity * finalData[i].price)
70      } else{
71          value[name.lastIndexOf(finalData[i].category)] += finalData[i].quantity * finalData[i].price
72      }
73  }
```

Figura 10: Consolidando valores por categoria.

Ao fim, foi criada outra constante **totalValueByCategory** que unifica as constantes **name** e **value** utilizando a função **forEach(),** após isso com as funções **JSON.stringify** e console.log() a constante totalValueByCategory foi convertida para JSON e impressa no terminal.

4. Resultado final: impressão do terminal


```
[
        {
                "id": 1911864,
                "name": "Mouse Gamer Predator cestus 510 Fox Preto",
                "price": 699,
                "category": "Acessórios",
                "quantity": 0
        },
        {
                "id": 9628920,
                "name": "Lava & Seca 10,2 Kg Samsung Eco bubble branca com 09 Programas de Lavagem",
                "quantity": 57,
                "price": 3719.7,
                "category": "Eletrodomésticos"
        },
        {
                "id": 1316334,
                "name": "Refrigerador bottom Freezer Electrolux de 02 Portas Frost Free com 598 Litr
os",
                "quantity": 12,
                "price": 3880.23,
                "category": "Eletrodomésticos"
        },
        {
                "id": 6502394,
                "name": "Fogão de Piso Electrolux de 04 bocas, Mesa de Vidro Prata",
                "quantity": 37,
                "price": 1419,
                "category": "Eletrodomésticos"
        },
        {
                "id": 9576720,
                "name": "Forno Micro-ondas Panasonic com capacidade de 21 Litros branco",
                "quantity": 13,
                "price": 358.77,
                "category": "Eletrodomésticos"
        },
        {
                "id": 8875900,
                "name": "Smart TV 4K Sony LED 65" 4K X-Reality Pro, UpScalling, Motionflow XR 240 e
Wi-F",
                "quantity": 0,
                "price": 5799.42,
                "category": "Eletrônicos"
        },
        {
                "id": 9746439,
                "name": "Home Theater LG com blu-ray 3D, 5.1 canais e 1000W",
                "quantity": 80,
                "price": 2199,
                "category": "Eletrônicos"
        },
        {
                "id": 2162952,
                "name": "Kit Gamer acer - Notebook + Headset + Mouse",
                "price": 25599,
                "category": "Eletrônicos",
                "quantity": 0
        },
        {
                "id": 3500957,
                "name": "Monitor 29 LG FHD Ultrawide com 1000:1 de contraste",
                "quantity": 18,
                "price": 1559.4,
                "category": "Eletrônicos"
        },
        {
                "id": 5677240,
                "name": "Conjunto de Panelas antiaderentes com 05 Peças Paris",
                "quantity": 21,
                "price": 192.84,
                "category": "Panelas"
        }
]
{
        "Acessórios": 0,
        "Eletrodomésticos": 315752.67000000004,
        "Eletrônicos": 203989.2,
        "Panelas": 4049.64
}
Corrected Data!
```


Figura 11: Impressão no terminal

Porque Javascript?

Por ser uma linguagem versátil agindo tanto no **front-end** como no **back-end**, o **JavaScript** é uma das linguagens mais populares atualmente. Por ser também de fácil aprendizado, ela é ótima para iniciantes, sem dizer também da facilidade de se tornar **fullStack** com ela.