



FULL DIGITAL PERFORMANCE

PROJETO CLASSIFICATÓRIO
WEB ANALYTICS

Candidato: Gabriel Antonio Fogaça Vieira

1. LEITURA DO ARQUIVO .JSON

Através da função “LerArquivo()”, o programa executa a função “readFileSync”, entregando assim uma estrutura de arrays criada a partir do arquivo .json.

```
function LerArquivo() {  
    var fs = require('fs');  
    // Leitura do arquivo JSON  
    var data = fs.readFileSync('./broken-database.json', 'utf8');  
    // Transforma os dados do arquivo em estrutura de array json  
    var dados = JSON.parse(data);  
  
    return dados  
}
```

```
var brokenBD = LerArquivo()
```

Aqui especifiquei o valor da variável brokenBD como o arquivo .json já transformado em array.

1.1. COMANDO FOR

A partir do comando “for” com o arquivo .json já convertido em array podemos “varrer” a estrutura buscando determinados elementos.

```
for (var i in brokenBD) {
```

2. RESOLUÇÃO DE PROBLEMAS

2.1. NOME

Para a correção dos caracteres especiais no elemento 'nome', foi criada uma função com os comandos `split()` e `join()`, eles foram escolhidos pois o comando `replace()` apresentou bugs quando o item possuía mais de um caracter a ser substituído.

```
// Função para correção dos nomes
function CorrigirNome(val) {

    var cRet = ''

    cRet = val.split("ø").join("o").split("æ").join("a").split("ç").join("c").split("ß").join("b")

    return cRet
}

// Executa a substituição dos caracteres especiais conforme solicitado
brokenBD[i]['name'] = CorrigirNome(brokenBD[i]['name'])
```

2.2. PRICE

Através de uma função criada para validar se a informação contida no elemento 'price' da array era um numero ou uma string, foi criado uma condição, cujo intuito era através do comando `parseFloat()` transformar a string em float/número

```
function Isnumber(val) {

    return typeof val == "number"
}

// Converte o conteudo da variavel Price para Float caso o conteudo seja string
if (!Isnumber(brokenBD[i]['price'])) {
    brokenBD[i]['price'] = parseFloat(brokenBD[i]['price'])
}
```

2.3. QUANTITY

Para corrigir o problema e adicionar o elemento quantity foi criada uma validação junto a uma breve função que era responsável por adicionar “quantity 0” na array.

```
// corrigir o problema do Quantity
if (typeof brokenBD[i]['quantity'] == 'undefined') {
    brokenBD[i] = quantityadd()
}

//função para adicionar o quantity
function quantityadd(){
    var quantadd = { id: brokenBD[i]['id'], name: brokenBD[i]['name'], price: brokenBD[i]['price'], category: brokenBD[i]['category'], quantity: 0 }
    return quantadd
}
```

2.4. SALVAR DADOS (SAIDA.JSON)

Com uma função parecida com a de ler arquivo, foi possível criar uma para salvar, através do comando “writeFileSync” indicando o diretório e o nome do arquivo, junto a variável do arquivo de saída.

```
// Salva Arquivo (a função utilizada é basicamente a de ler o arquivo com algumas alterações como ao inves de readfile usamos o writefile)
function SalvarArquivo(oObj) {

    var fs = require('fs');
    // Leitura do arquivo JSON

    var dados = JSON.stringify(oObj);

    fs.writeFileSync('./saida.json', dados);
    // Transforma os dados do arquivo em estrutura de array json
}
```

```
// Exportar o novo JSON com os dados do array brokenBD

SalvarArquivo(brokenBD)
```

3. VALIDAÇÃO DO .JSON

Através da função de ler arquivo criada anteriormente, fiz o input do `saida.json` para a validação dos dados, com o comando “`readfilesync`”.

3.1. ORDENAÇÃO POR CATEGORIA E ID

A ordenação por ordem alfabética no caso da categoria e numérica no caso do ID, foi realizado através de uma função de validação de parâmetro (a, b), com isso o através do return o a função decide se a ordem de x objeto é antes ou depois do próximo objeto. Primeiro foi realizado a ordem em categorias, logo em seguida ordenado por id.

Para imprimir os dados de saída gerados foi necessário um comando `.stringify(json)`, transformado na função “`imprimirJSON`” para facilitar a utilização.

```
// Ordenação por Categoria em ordem alfabetica e logo em seguida por ID
BDsaida.sort(function (a, b) {
    if (a.category > b.category) {
        return 1
    }
    if (a.category < b.category) {
        return -1
    }
    if (a.id > b.id) {
        return 1
    }
    if (a.id < b.id) {
        return -1
    }
    return 0
})

console.log (BDsaida)

for (var i in BDsaida) {
    imprimirJSON(BDsaida[i]['category'])
    imprimirJSON(BDsaida[i]['name'])
    imprimirJSON(BDsaida[i]['id'])
}
```

```
//Transformar em dados de saída arquivos em JSON
function imprimirJSON(json) {
  console.log(JSON.stringify(json));
}
```

3.2. VALOR TOTAL DO ESTOQUE

Para realizar a soma do estoque e facilitar possíveis alterações futuras, adicionei uma função para buscar no array a categoria, caso ela não tenha sido encontrada o valor retornado seria -1, e através do uso do “if” e “else” validei as informações, utilizei do comando .push() para adicionar o elemento na array “categoria” caso ele ainda não exista.

```
var categoria = [];
var nret = 0;

for (i in BDsaida) {
  nret = Search(categoria, BDsaida[i]['category']);
  if (nret >= 0) {
    categoria[nret]['price'] += BDsaida[i]['price'] * BDsaida[i]['quantity']
  }
  else {
    categoria.push({ category: BDsaida[i]['category'], price: BDsaida[i]['price'] * BDsaida[i]['quantity'] })
  }
  //Validação para adicionar a soma de categoria junto a validação de que possui o produto em estoque
}
console.log(' ')
console.log('Total por categoria')
console.log(' ')

for (i in categoria) {
  imprimirJSON(categoria[i]['category'] + ' ' + categoria[i]['price'].toLocaleString('pt-br', {style: 'currency', currency: 'BRL'}))
}

//Função que localiza um elemento dentro da array
function Search(vetor, elemento) {

  var nret = -1
  //função dentro da array para localizar a posição do elemento passado como parametro
  nret = vetor.map(function (e) { return e.category; }).indexOf(elemento)

  return nret
}
```

4. JAVASCRIPT

Acredito que para trabalhar com edição e validação de arquivos .json o JavaScript seja uma boa opção, por ter uma boa “compatibilidade” com esse tipo de arquivo, por esse motivo foi a linguagem escolhida para o processo.

Outro ponto acredito que seja a versatilidade do JavaScript, já que ele tem um leque de funcionalidades, sendo uma das principais linguagens utilizadas no desenvolvimento web.