
# Projeto Classificatório

Processo seletivo - Web Development

# Documentação da Resolução do Projeto

## Recuperação dos dados originais do banco de dados

Para recuperação dos dados originais iniciei importando o módulo ‘fs’ File System que já é nativo no Node.js e na sequencia declarei uma função com o nome “lerJson” para ler o arquivo broken-database.json que irá simular nosso banco de dados.

```
const fs = require("fs");

function lerJson() {
  const data = fs.readFileSync("./broken-database.json", "utf-8");
  return JSON.parse(data);
}
```

Essa função consiste em armazenar dentro da variável “data” o conteúdo do arquivo JSON que é passado como STRING, para podermos transforma-la em um objeto retornaremos a mesma variável “dados” convertida em JSON resultando assim em um objeto que acessaremos mais tarde nas próximas funções.

Para saber mais sobre as diversas funcionalidades úteis para acessar e interagir com o file system acesse Módulo fs do Node.js.

Já com a primeira função que ler o arquivo JSON criada poderemos dar sequência em nosso algoritmo e começar a tratar os erros presentes no banco de dados.

Para ter acesso ao conteúdo do banco de dados, criei uma variável com nome “dados” e atribuir a ela a nossa primeira função que nos retorna o conteúdo do arquivo JSON convertido em objeto.

```
let dados = lerJson();
```

Tendo acesso ao conteúdo criei a segunda função que tem como seu objetivo percorrer o conteúdo do arquivo (objeto) e sua propriedade “name” com o método forEach() e replace() para encontrar e substituir " æ " por "a", "¢" por "c", " ø " por "o", " ß " por "b” com auxílio de uma expressão regular (RegExr), e como retorno da função os nomes corrigidos.

```
function corrigirNomes() {
  dados.forEach((elemento) => {
    elemento.name = elemento.name
      .replace(/(æ)/g, "a")
      .replace(/(¢)/g, "c")
      .replace(/(ø)/g, "o")
      .replace(/(ß)/g, "b");
  });
  return dados;
}
```

Para saber mais sobre manipulação de string com replace e regex acesse JavaScript replace: manipulando Strings e regex.

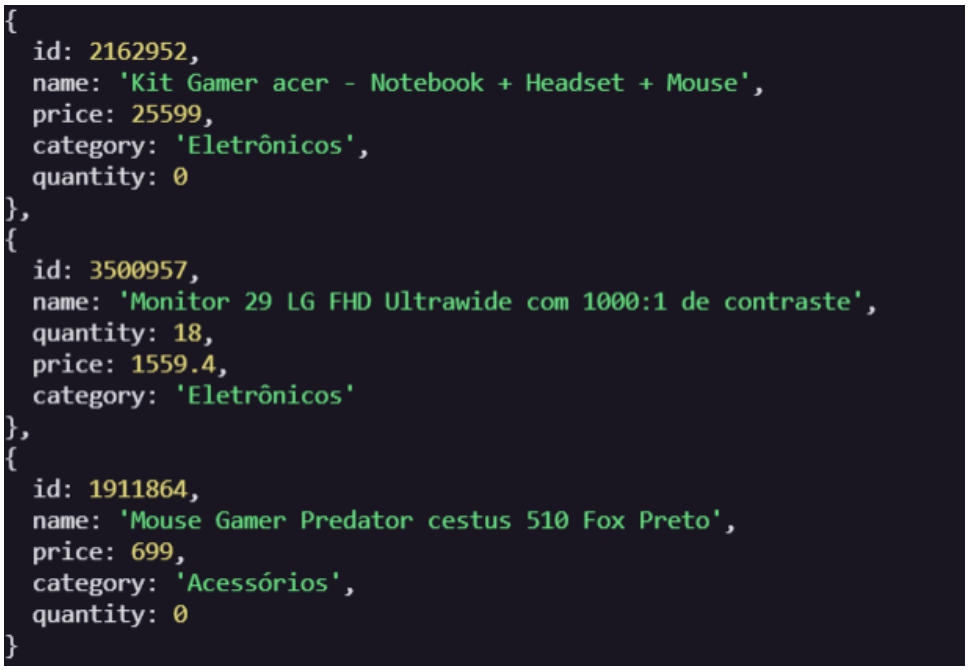
Para saber mais sobre o método forEach() acesse forEach()

Partindo agora para terceira função do nosso algoritmo declarei uma função chamada “corrigirPrecos” que irá percorrer os elementos do (Objeto) e encontra o tipo deles e caso eles tenham o seu tipo como STRING os mesmos serão convertidos em NUMBER.

```
function corrigirPrecos() {
  dados.forEach((elemento) => {
    if (typeof elemento.price === "string") {
      elemento.price = Number(elemento.price);
    }
  });
  return dados;
}
```

Chegamos a última função de correção dos dados corrompidos, iniciei declarando a função com o nome “corrigirQuantidades” que tem como objetivo percorrer a propriedade “quantity” do (Objeto) com o método forEach() para verificar se o (Objeto) contém a propriedade “quantity”, se for verificado que o (Objeto) não contem a propriedade “quantity” entrara no IF(condicional) e adicionara a propriedade “quantity” com valor 0.

Passando como retorno da função o **console,log(dados)** notaremos que a saída no terminal tem o (Objeto) com a propriedade ‘quantity” mas não na ordem correta, conforme o exemplo original a propriedade “quantity” vem logo após a propriedade “name” e nesse resultado onde não tinha a propriedade “quantity” agora foi adionada na última posição.



Para essa segunda parte da função onde iremos corrigir a ordenação das propriedades do (Objeto), iniciei declarando uma constante e atribuindo a ela um (Objeto) com as propriedades semelhantes à nossos “dados” e em seguida atribuir a variável “dados” o resultado do algoritmo que ordenar as propriedades na ordem correta usando o método **map()**, ele faz a leitura de todos os elementos do array de (Objeto), executa uma função callback para cada um e devolve como retorno um novo array de (Objeto) e dentro da callback usei o **Obect.assign()** que nos permite copiar os valores de um ou mais objetos e passar para um outro, passando assim como primeiro parâmetro um objeto vazio e um segundo parâmetro objeto.keys(elemento) que retorna um array de propriedades enumeráveis de um determinado objeto, na mesma ordem em que é fornecida e em seguida usei o método **sort()** que permite classificar (ordenar) elementos de um *array* local. Além de retornar o *array* ordenado, o método sort() altera as posições dos elementos no *array* original e para finalizar esta segunda parte temos como o retorno da função a variavel “dados” já com todo seu conteúdo corrigido e ordenado.

```
function corrigirQuantidades() {
  dados.forEach((elemento) => {
    if (!elemento.quantity) {
      elemento.quantity = 0;
    }
  });
  const sortOrder = { id: 1, name: 2, quantity: 3, price: 4, category: 5 };
  dados = dados.map((elemento) =>
    Object.assign(
      {},
      ...Object.keys(elemento)
        .sort((a, b) => sortOrder[a] - sortOrder[b])
        .map((indice) => {
          return { [indice]: elemento[indice] };
        })
    )
  );
  return dados;
}
```

Agora para última função da recuperação dos dados originais iniciei uma constante “data” e atribuir a ela o valor do conteúdo da variável “dados” convertido em STRING e formatado conforme o arquivo JSON, com módulo File System ‘fs’ criei um novo arquivo chamado “saida,json” como primeiro parâmetro e como segundo parâmetro constante data com os dados convertidos em STRING e formatado tendo assim como resultado um novo arquivo JSON com os dados recuperados e corrigidos.

```
function exportarJson() {
  const data = JSON.stringify(dados, null, 2);
  fs.writeFileSync("./saida.json", data);
}
```

## Validação do banco de dados corrigido

Para função que imprime a lista com todos os nomes dos produtos, ordenados primeiro por categoria em ordem alfabética e ordenados por id em ordem crescente iniciei declarando uma função com o nome "imprimirNomes" e passando como parâmetro um input que será nosso arquivo JSON corrigido em seguida declarei uma lista vazia com nome "lista" e usei novamente o método **forEach()** para percorrer o (Objeto) e com o método **sort()** ordenar **a** lista conforme pedido na resolução o método sort() nesta função irá pegar os nomes das categorias do (Objeto) e compara-los conforme a condição imposta e irar ordena-lo em ordem alfabética e em seguida ordenara pelo ID em ordem crescente e para cada item percorrido irá dar um lista,push na propriedade "name" para adicionar o nome do produto na lista, resultando assim em uma lista ordenada por ordem alfabética e por id em ordem crescente.

```
function imprimirNomes(input) {
  let lista = [];
  input.forEach((elemento) => {
    input.sort((a, b) => {
      if (a.category.toLowerCase() > b.category.toLowerCase()) return 1;
      else if (a.category.toLowerCase() < b.category.toLowerCase()) return -1;
      else if (a.id < b.id) return -1;
      return 0;
    });
    lista.push(elemento.name);
  });
  return console.log(lista);
```

Para segunda e última função da validação do banco de dados iniciei declarando uma função com o nome "valorTotalPorCategoria" e passando como parâmetro um input que será nosso arquivo JSON corrigido e em seguida declarei 4 variáveis com valor 0 que representara mais tarde o valor final do estoque por categoria e iniciei um loop for para percorrer cada propriedade "category" e conforme a condicional atribuir a variável totalEstoquecategoria"x" pela expressão aritmética (totalEstoquecategoria"X" + (Proriedade"price" * propriedade"quantity) da determinada categoria resultando assim em um valor total do estoque por categoria, ou seja, a soma do valor de todos os produtos em estoque de cada categoria, considerando a quantidade de cada produto.

```
function valorTotalPorCateegoria(input) {
  let totalEstoqueAcessorios = 0;
  let totalEstoqueEletrodomesticos = 0;
  let totalEstoqueEletronicos = 0;
  let totalEstoquePanelas = 0;

  for (let i = 0; i < input.length; i++) {
    if (input[i].category === "Acessórios") {
      totalEstoqueAcessorios += input[i].price * input[i].quantity;
    } else if (input[i].category === "Eletrodomésticos") {
      totalEstoqueEletrodomesticos += input[i].price * input[i].quantity;
    } else if (input[i].category === "Eletrônicos") {
      totalEstoqueEletronicos += input[i].price * input[i].quantity;
    } else if (input[i].category === "Panelas") {
      totalEstoquePanelas += input[i].price * input[i].quantity;
    }
  }
  return console.log(`
${totalEstoqueAcessorios.toFixed(2)},
${totalEstoqueEletrodomesticos.toFixed(2)},
${totalEstoqueEletronicos.toFixed(2)},
${totalEstoquePanelas.toFixed(2)},
`);
}
valorTotalPorCateegoria(dados);
```