Projeto Classificatório

Processo seletivo - Web Analytics

Documentação

Teste Prático

Michelle Ghiraldi de Castro

Outubro/2021

Neste documento explico e mostro as funcionalidades presentes em meu algoritmo.

1. **Recuperação dos dados originais do banco de dados**

O primeiro passo para montar o algoritmo foi a leitura do arquivo **broken-database.json,** para isso, foi utilizado o método ***'require',*** método tal que carrega um arquivo ou biblioteca através de um dado caminho. Segue código abaixo:

```
var json = require('./broken-database.json');
```

Logo após foram criadas as funções de correção dos dados do arquivo **json**, seguem funções abaixo:

- **changeLetters** -> função que corrige as letras. É possível ver que a mudança das letras foi feita através do método ***Regex***, utilizado para substituir caracteres de uma palavra, e do modificador ***'g'***, que indica que todas as ocorrências do texto buscado sejam substituídas.

```
function changeLetters(jsonText) {

    var jsonUpdated = jsonText.replace(new RegExp('æ', 'g'),'a');
    jsonUpdated = jsonUpdated.replace(new RegExp('¢', 'g'),'c');
    jsonUpdated = jsonUpdated.replace(new RegExp('ø', 'g'),'o');
    jsonUpdated = jsonUpdated.replace(new RegExp('ß', 'g'),'b');

    return jsonUpdated;
}
```

- **changePrices** -> função que corrige os preços. É possível ver que foi utilizado o comando condicional ***if*** e o operador ***typeof*** para selecionar os preços que estavam em formato **string,** e a função ***parseFloat()*** para convertê-los em números reais.

```
function changePrices(product){
    if(typeof product.price === 'string'){
        product.price = parseFloat(product.price);
    }
    return product;
}
```

- **addQuantity** -> função que adiciona valor "0" a propriedade quantidade. É possível ver que foi utilizado novamente o comando condicional ***if*** e a função ***hasOwnProperty*** para verificar se haveria a propriedade quantity. Caso inexistente, o programa insere a propriedade e o valor "0", como visto abaixo.

```
function addQuantity(product){
    if(!product.hasOwnProperty("quantity")){
        product.quantity = 0;
    }
    return product;
}
```

- **exportDatabase** -> função que exporta o arquivo JSON com o banco corrigido. Abaixo, o método ***'require'*** usado para importar a biblioteca.

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

Para a criação deste arquivo foi utilizado o módulo ***fs*** (file system) e o método ***writeFile*** que cria o arquivo saida.json, utilizando a codificação 'utf8'. Também é utilizada uma função ***err*** para caso haja algum erro, o sistema irá informar o usuário caso seu arquivo não seja exportado corretamente.

```
function exportDatabase(jsonData){

    fs.writeFile("saida.json", jsonData, 'utf8', function (err) {
        if (err) {
            console.log("Houve um erro na exportação do arquivo JSON.");
            return console.log(err);
        }
        console.log("O arquivo JSON foi salvo.");
    });
}
```

2. **Validação do banco de dados corrigido**

Encerradas as funções de recuperação e correção de dados, aqui iniciam-se as funções para a validação do banco de dados corrigido:

- **orderedDatabase** -> função que ordena as categorias alfabeticamente e o id em ordem crescente. É possível ver que foi utilizado o método ***sort***, que ordena elementos de uma ***array*** e retorna a ***array***, também foi utilizado o comando condicional ***if***, que verifica se as categorias possuem o mesmo valor e as organiza em ordem crescente (***return -1***), do menor para o maior. Observe abaixo:

```
function orderedDatabase(jsonConverted){
    //trecho copiado e adaptado de https://gomakethings.com/sorting-an-
array-by-multiple-criteria-with-vanilla-javascript/
    jsonConverted.sort(function ( a, b){
        if(a.category < b.category) {
            return -1;
        } else {
            if ((a.category == b.category) && (a.id < b.id)) {
                return -1;
            }
            return true;
        }
```

```
    });
    console.log(jsonConverted);
    return jsonConverted;
}
```

- **totalSum** -> função que soma o valor total do estoque por categoria. Para tal, foi utilizado o método ***forEach*** e o comando condicional ***if***, que verificam se existe repetição de categorias dentro do banco de dados, caso sim, é feito o cálculo do estoque de cada produto e logo após é feita a soma dos que possuem a mesma categoria. Como visto no código abaixo:

```
function totalSum(orderedJson){
    var totalSum = 0, category = "", oldCategory = "";
    orderedJson.forEach(element => {
        category = element.category;
        if(category == oldCategory || oldCategory == ""){
            totalSum = totalSum + element.price * element.quantity;
        } else {
            console.log(oldCategory, totalSum);
            totalSum = element.price * element.quantity;
        }
        oldCategory = category;
    });

}
```