Projeto Classificatório ROCKY

Processo seletivo - Web Analytics

Gabriel Nunes de Moraes Ghirardelli

Sorocaba, 08 de Janeiro de 2022

Para cada uma das etapas propostas e listadas abaixo, foi criado uma função, as quais terão uma breve descrição no decorrer deste documento.

Sumário

1. Recuperação dos dados originais do banco de dados

1.1 Leitura do arquivo Json

Esta primeira função reliza o requerimento do arquivo que contém os dados corrompidos, e retorna os dados para a chamada da função. Caso o arquivo não seja encontrados, a função retorna uma mensagem informando.

```
function readJson(){
    try {
        return require('./broken-database.json')
    } catch (MODULE_NOT_FOUND){
        console.log("File not found");
        return null;
    }
}
```

1.2 Correção dos nomes

Para cada um dos registros do conjunto de dados, esta função recupera o valor corrompido, realiza as substituições e registra o valor corrigido na base de dados.

```
function fixName(data){
    data.forEach(obj => {
        let name = obj.name
        name = name.replace(/æ/g,"a")
        name = name.replace(/¢/g,"c")
        name = name.replace(/ø/g,"o")
        name = name.replace(/ß/g,"b")
        obj.name = name
    });
    return
}
```

1.3 Correção dos preços

Para cada um dos registros do conjunto de dados, esta função verifica se o registro é do tipo string, e caso necessário, recupera o valor convertendo-o para o devido formato e registra o valor corrigido na base de dados.

```
function fixPrice(data){
    data.forEach(obj => {
        if(typeof obj.price == "string"){
            let price = parseFloat(obj.price)
            obj.price = price
        }
    });
    return
}
```

1.4 Correção das quantidades

Para cada um dos registros do conjunto de dados, esta função verifica se o atributo quantity existe no registro que está sendo verificado, e caso o mesmo não exista, o atributo é criado com o valor 0.

```
function fixQuantity(data){
    data.forEach(obj => {
        if(obj.quantity == undefined){
            obj.quantity = 0
        }
    });
    return
}
```

1.5 Exportar um arquivo JSON com o banco corrigido

Após as correções dos dados, esta função converte a base de dados para o formato necessário, e em seguida cria o arquivo saida.json, que contém os dados recuperados.

```
function exportSolutionToJson(data) {
    let json = JSON.stringify(data)
    let fs = require('fs')
    fs.writeFile('saida.json',json, (err) => {
        if (err) console.log(err)
        console.log('\nSolution file created')
    })
    return
}
```

2. Validação do banco de dados corrigido

2.1 Lista ordenada

Essa função copia os registros da base de dados utilizando-se da função sort() em conjunto com uma função de comparação, as quais realizam a ordenação dos dados de acordo com sua categoria e pelo seu id. Após a ordenação dos produtos, a lista ordenada é impressa no terminal.

```
function orderedList(data) {
    let list = data.sort( function(a,b) {
        if (a.category > b.category) {
            return 1;
        } else if (a.category < b.category) {
            return -1;
        } else {
            if (a.id > b.id) {
                return 1;
            } else if (a.id < b.id) {
                return -1;
            } else {
                return 0;
            }
        }
    });
    console.log(list);
    return;
}
```

2.2 Estoque por categoria

Para cada um dos registros do conjunto de dados, essa função verifica se a categoria já foi inclusa na listagem final, e caso necessário inclui a mesma. Em seguida, adiciona o valor de estoque do produto atual, ao valor total da categoria. Após a inclusão de todos os produtos, a listagem de valor por categoria é impressa no terminal.

```
function valuePerCategory(data) {
    let categories = []
    let values = []
    data.forEach(obj => {
        if (categories.indexOf(obj.category) == -1) {
            categories.push(obj.category)
            values.push(0)
        }
        values[categories.indexOf(obj.category)] += obj.quantity * obj.price
    });

    for(let i = 0; i < categories.length; i++){
        console.log(categories[i] + ": " + values[i].toFixed(2))
    }

    return;
}
```

3. Execução das funções

Este trecho final do código corresponde as chamadas de todas as funções mencionadas, assim realizando todo o processo de recuperação e validação propostos.

```
const data = readJson();
if (data == null) {
    return;
}
fixName(data);
fixPrice(data);
fixQuantity(data);
exportSolutionToJson(data);

console.log("----------- Ordered list ------------")
orderedList(data);

console.log("\n-------- Value per category ---------")
valuePerCategory(data);
```