DESENVOLVIDO POR GABRIEL HENRIQUE MOREIRA ALVES

PROGRAMA CORRETOR DE CASO ESPECÍFICO

DATA:22/06/2021

SUMÁRIO

INTRODUÇÃO

Este código tem como objetivo principal recuperar e retornar os dados corrompidos de um banco de dados ao formato esperado pelo sistema, uma vez que existem 3 tipos de irregularidade nesses dados.

A primeira irregularidade detectada nos dados foi a alteração de caracteres que compõem os nomes de produtos específicos, todos os caracteres “a” foram substituídos por “æ”, os “c” se tornaram “¢”, os caracteres “o” por “ø” e por fim todos os “b” foram substituídos por “ß”.

Como segunda irregularidade nos dados temos algumas alterações nos tipos da propriedade de preço, sendo ele trocado do tipo *number para o formato de uma **string.

O último problema acarretado pelo corrompimento do banco de dados é o desaparecimento de algumas propriedades “quantity” onde o valor atribuído era 0.

---

*Formato numérico de uma variável em programação

*Formato de texto de uma variável em programação

LINGUAGEM UTILIZADA

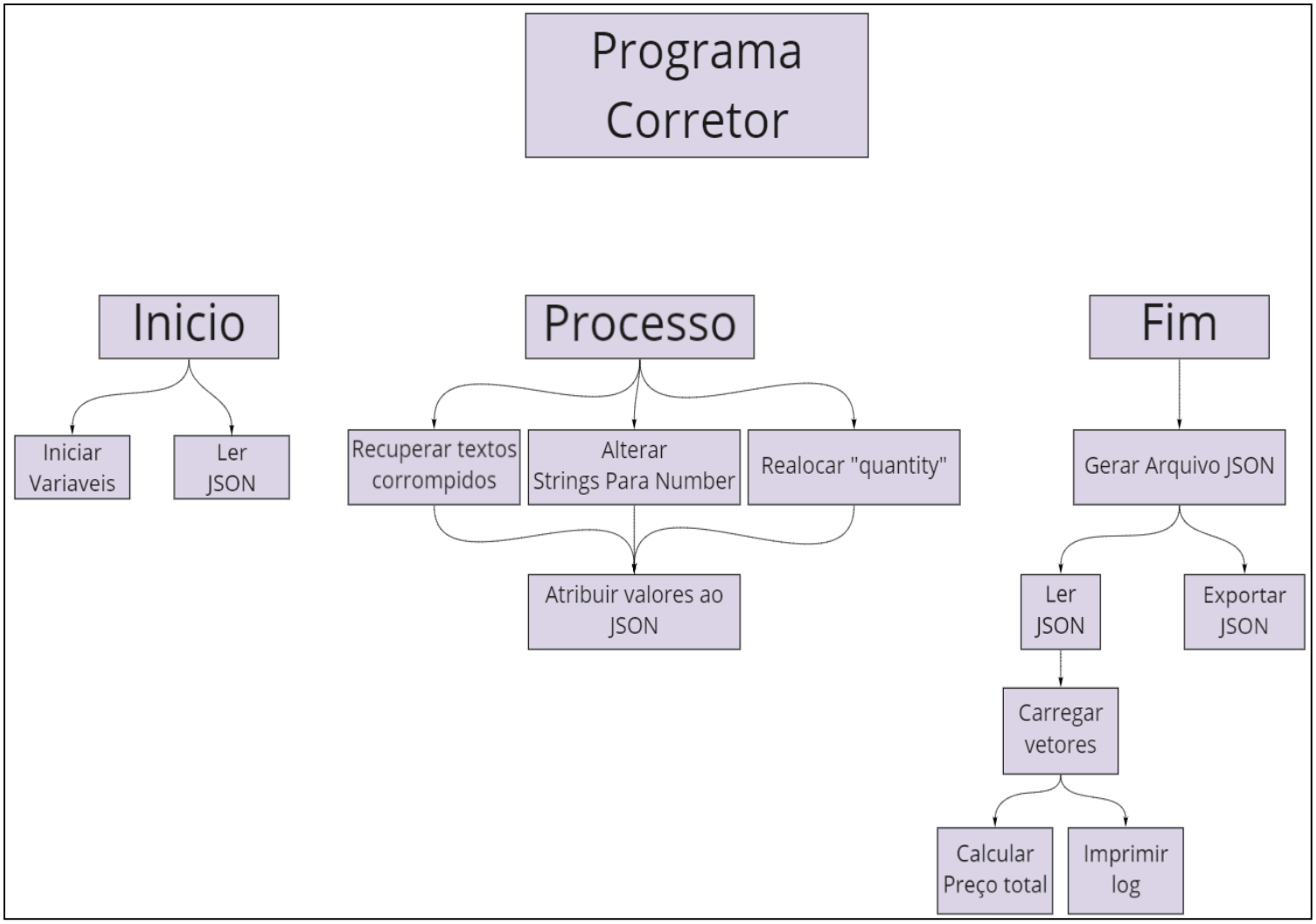
O programa foi desenvolvido através da linguagem de programação Java script, utilizando-se de conceitos do paradigma de programação estruturada, pois ele se demonstra o mais adequado a este projeto de baixa complexidade, onde apenas algumas funções simples são necessárias para seu funcionamento.

A linguagem Java script foi empregada neste projeto pois ela carrega consigo uma boa portabilidade do projeto, uma vez que ela é a linguagem mais popular no mercado de desenvolvimento web atualmente, junto a isso é uma linguagem de uso relativamente simples, sendo assim possível a manutenção do software sem a necessidade de um especialista da área.

DIAGRAMA HIERÁRQUICO DAS FUNÇÕES

Este projeto possui 3 funções principais que corrigem o arquivo JSON utilizado como entrada de dados, essas funções podem ser observadas no seguinte diagrama:

Figura 1-Diagrama de funções

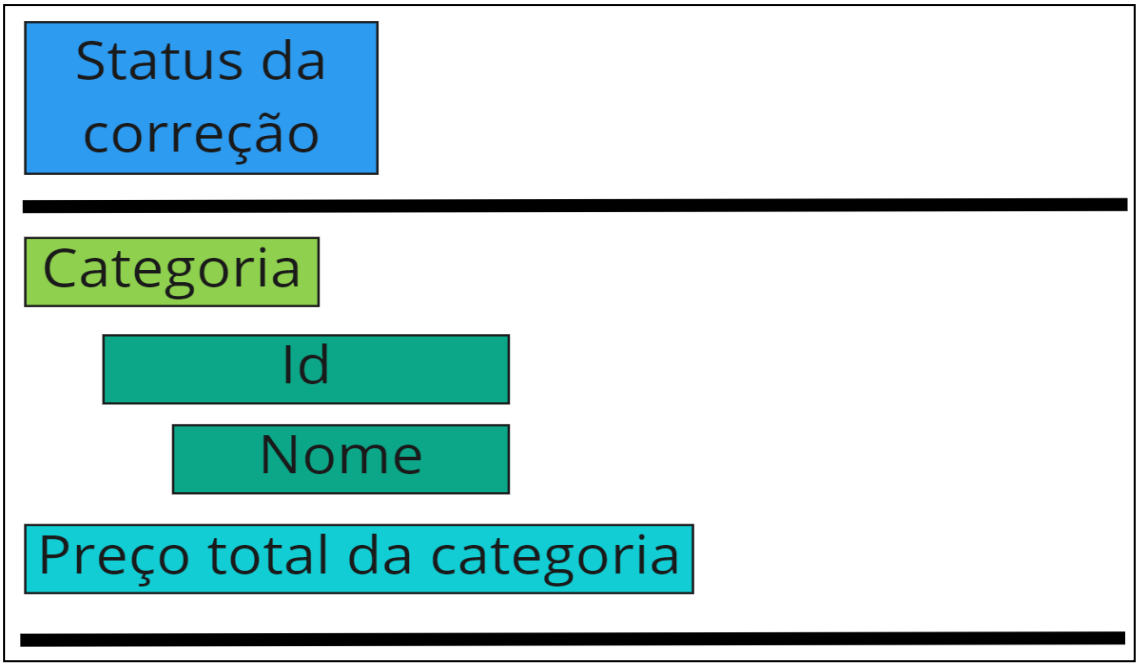


Autoria própria

INTERFACE DE VALIDAÇÃO

A fim de verificar o êxito da correção do arquivo corrompido, o programa imprime no log do sistema informações lidas do arquivo corrigido, essas informações são informadas de acordo com o seguinte layout:

Figura 2- Layout do log



Na secção “Status da correção” será exibido se o arquivo foi corrigido ou se ocorreu algum tipo de erro durante o processo, após isso será exibido em ordem alfabética as categorias contidas no arquivo do banco de dados corrigido.

Dentro de cada categoria por sua vez serão apresentados todos os Ids de produtos da categoria em ordem crescente, assim como os nomes correspondentes dos ids dos produtos.

Por fim ao final de todos os itens da categoria, será exibido o valor de todos os produtos da categoria levando em consideração a quantidade atual do produto em estoque.

FUNCIONALIDADES DO PROGRAMA

Este programa possui 5 funções ao todo, sendo 3 delas exclusivas para a correção do banco de dados corrompidos, essas funções corrigem as trocas de caracteres, tipos errados de dados, e propriedades ausentes, tais funções podem ser observadas a seguir:

```
function correcaoTextual(texto) //Função que corrige os textos corrompidos
{
    var tamanhoTexto=texto.length;
    for (var i = 0; i < tamanhoTexto; i++)
    {
        if (texto[i]== "æ")
        {
            texto=texto.replace('æ','a')
        }
        else if (texto[i]== "¢")
        {
            texto=texto.replace('¢','c')
        }
        else if (texto[i]== "ø")
        {
            texto=texto.replace('ø','o')
        }
        else if (texto[i]== "ß")
        {
            texto=texto.replace('ß','b')
        }
     }
    return texto
}
```

Esta função localiza os caracteres corrompidos dentro de uma string, e os corrigepara os valores originais. Após isso retorna o valor para ser atualizado no arquivo. JSON.

```
function correcaoDePrecos() //Função que corrige os preços que estão como string
{

    try {
        if (numero!=Number)
        {
            numero= parseFloat(numero)
        }
        else
        {
            numero=numero
        }
        return numero

     }
     catch (e)
     {
        console.log("ERRO AO CORRIGIR VALORES DE STRING PARA NUMBER")
     }

}
```

Esta é a segunda função corretora do programa, ela verifica se o valor carregado na variável número é de um tipo distinto de numérico, em caso positivo ele atualiza número para o tipo numérico caso negativo o número continua sendo ele mesmo.

```
function correcaoDeQuantidades() //Função que quantidades que foram apagadas
{
   if (quantidade==undefined)
   {
        quantidade= 0
   }
   else
   {
    quantidade=quantidade
   }
    return quantidade
}
```

Por fim como 3ª função corretora esta função verifica se a propriedade “quantity” não está presente, caso positivo ele recria a propriedade e atribui o valor 0 a ela, em caso negativo ela continua sendo ela mesma.

A aquisição dos dados para o retrabalho pode ser observada no seguinte trecho de código:

```
function ler(nomeArquivo) //Função para ler arquivo
{
    try
    {
        var obj=require(nomeArquivo)
        return obj
    }
    catch (e)
    {
        console.log("ERRO AO LER ARQUIVO JSON")
    }
}
```

A função ler() irá ler o arquivo JSON e logo em seguida irá transformar os dados do arquivo em um objeto para poder ser trabalhado em cima desses dados.

Após isso em um ciclo de repetição as 3 funções serão chamadas usando como parâmetro os valores contidos no objeto do arquivo json.

```
function executarCorrecao(arquivo)
{
    for(k=0;arquivo[k]!=null;k++)
    {
        arquivo[k].price=correcaoDePrecos(arquivo[k].price)
        arquivo[k].name=correcaoTextual(arquivo[k].name )
        arquivo[k].quantity=correcaoDeQuantidades(arquivo[k].quantity)
    }
}
```

Depois de todos os dados serem corrigidos pelas 3 funções corretoras o programa ira gerar um novo arquivo JSON com os valores corrigidos, isso pode ser observado no seguinte trecho de código:

```
function escrever(arquivoNome,conteudo) //Função para escrever arquivo e validar
{
    try
    {
        json = JSON.stringify(conteudo) //retorna o para json
        fs.writeFile(arquivoNome, json,function(err) //Escreve um arquivo
        {
            console.log("Arquivo Corrigido")//Imprime que o arquivo foi corrigido
            imprimir()//Chama Validaçao

        })
    }
    catch (e)
    {
        console.log("ERRO AO ESCREVER ARQUIVO JSON")
    }
}
```

Após todo esse processo de correção como forma de validação, o arquivo gerado como saída será lido pelo programa, e os dados lidos serão carregados a vetores que após um processo de organização será impresso no terminal as informações contidas no documento. Essa funcionalidade esta presente no seguinte trecho de código:

```
arquivo=ler('./saida.json')

 for(p=0;arquivo[p]!=null;p++)
     {
         if(arquivo[p].category=="Acessórios")          //Verifica se a categoria observada é de Acessórios
             {
                 posicaoDosAcessorios.push(p)           //Guarda quais posicoes fazem parte da categoria Acessórios
             }
         else if (arquivo[p].category=="Eletrônicos")   //Verifica se a categoria observada é de Eletrônicos
             {
                 posicaoDosEletronicos.push(p)          //Guarda quais posicoes fazem parte da categoria Eletrônicos
             }
         else if(arquivo[p].category=="Eletrodomésticos")//Verifica se a categoria observada é de Eletrodomésticos
             {
                 posicaoDosEletrodomesticos.push(p)     //Guarda quais posicoes fazem parte da categoria Eletrodomésticos
             }
         else if(arquivo[p].category=="Panelas")        //Verifica se a categoria observada é de Panelas
             {
                 posicaoDasPanelas.push(p)              //Guarda quais posicoes fazem parte da categoria Panelas
             }
     }
```

As posições do índice dos produtos de cada categoria estão sendo guardadas para posteriormente poderem ser chamadas de forma ordenada por categorias.

```
idsAcessorios=idsAcessorios.sort()  // Ordena em ordem crescente as ids dos acessorios
idsEletronicos.sort()               // Ordena em ordem crescente as ids dos eletronicos
idsEletrodomesticos.sort()          // Ordena em ordem crescente as ids dos eletrodomesticos
idsPanelas.sort()                   // Ordena em ordem crescente as ids das panelas
```

Neste trecho as Ids de cada categoria estão sendo organizadas em ordem crescente.

A seguinte função demonstra como funciona a impressão final no log das informações que serão utilizadas como forma de validação.

```
function validacao(n,m,catI,posicao,objL,idsCat,obId,oNome,oPreco,oQuantidade)// Função que imprime log de validação
{
    try
    {
        console.log("_____________________________________________________________________________")
        console.log(catI)
        for(n=0;n<posicao.length;n++)
        {
            while(m<objL.length)
            {
                if(idsCat[n]==obId[m])
                {
                    console.log("   Id:"+idsCat[n])
                    console.log("       "+oNome[m])
                    somaDePreco(oPreco[m],oQuantidade[m])
                }
                m++
            }
            m=0
        }
        console.log("Preço Total da Categoria:"+precototal)
        precototal=0
    }
    catch (e)
    {
        console.log("ERRO AO IMPRIMIR VALIDAÇÂO")
    }

}
```

Conforme observado na estrutura acima, todos os elementos de uma categoria serão percorridos para que quando um id seja encontrado na ordem correta de impressão, ele imprime o id e o nome correspondente a ele no log, junto a isso é calculado o valor total de todos os produtos no estoque daquela categoria.

Chamando a seguinte função de calculo:

```
function somaDePreco(numero,quantidade)//Função que soma todos o valor total de produtos de uma categoria
{
    try
    {
        precototal=precototal+numero*quantidade
    }
    catch (e)
    {
        console.log("ERRO AO CALCULAR PREÇO TOTAL DE CATEGORIA")
    }
    return precototal

}
```

TRATAMENTO DE ERROS

Para este projeto foram implementados em 6 pontos do código prevenções de erros.

Ao utilizar a função de ler um arquivo o try catch se faz necessário para o caso de alguma complicação ao ler o arquivo, poio aquivo pode estar em um formato inesperado e até mesmo corrompido.

Isso é empregue novamente quando o programa tenta exportar e reler o arquivo corrigido, pois novamente algum imprevisto ocorrer quando arquivos são lidos ou escritos.

O próximo possível erro previsto está presente na função corretora dos tipos number,pois pode ocorrer um erro caso um valor diferente de um número seja encontradona propriedade “price” do arquivo JSON.

Existem também tratamentos previstos para as funções de validação, sendo elas o cálculo de mercadoria por categoria e também para a impressão final no log.

O tratamento no cálculo de mercadorias existe para o caso de algum valor não esperado pelo sistema estar presente nos locais destinados a preço e quantidade.

E por fim o tratamento da função de impressão no log esta presente para casos em que as informações contidas no objeto lido não foram devidamente organizadas para a impressão.

Esses tratamentos podem ser observados nos seguintes trechos de código:

```
function ler(nomeArquivo) //Função para ler arquivo
{
    try
    {
        var obj=require(nomeArquivo)
        return obj
    }
    catch (e)
    {
        console.log("ERRO AO LER ARQUIVO JSON")
    }
}
```

```
function escrever(arquivoNome,conteudo) //Função para escrever arquivo e validar
{
    try
    {
        json = JSON.stringify(conteudo) //retorna o para json
        fs.writeFile(arquivoNome, json,function(err) //Escreve um arquivo
        {
            console.log("Arquivo Corrigido")//Imprime que o arquivo foi corrigido
            imprimir()//Chama Validaçao

        })
    }
    catch (e)
    {
        console.log("ERRO AO ESCREVER ARQUIVO JSON")
    }
}
```

```
function correcaoDePrecos(numero) //Função que corrige os preços que estão como string
{
    try
    {
        if (numero!=Number)
        {
            numero= parseFloat(numero)
        }
        else
        {
            numero=numero
        }
        return numero
    }
    catch (e)
    {
        console.log("ERRO AO CORRIGIR VALORES DE STRING PARA NUMBER")
    }
}
```

```
function somaDePreco(numero,quantidade)//Função que soma todos o valor total de produtos de uma categoria
{
    try
    {
        precototal=precototal+numero*quantidade
    }
    catch (e)
    {
        console.log("ERRO AO CALCULAR PREÇO TOTAL DE CATEGORIA")
    }
    return precototal
}
```

```
function validacao(n,m,catI,posicao,objL,idsCat,obId,oNome,oPreco,oQuantidade)// Função que imprime log de validação
{
    try
    {
        console.log("_____")
        console.log(catI)
        for(n=0;n<posicao.length;n++)
        {
            while(m<objL.length)
            {
                if(idsCat[n]==obId[m])
                {
                    console.log("   Id:"+idsCat[n])
                    console.log("       "+oNome[m])
                    somaDePreco(oPreco[m],oQuantidade[m])
                }
                m++
            }
            m=0
        }
        console.log("Preço Total da Categoria:"+precototal)
        precototal=0
    }
    catch (e)
    {
        console.log("ERRO AO IMPRIMIR VALIDAÇÂO")
    }
}
```