João Luiz da Costa Dias

DOCUMENTAÇÃO
Projeto classificatório Rocky Monks.

Objetivo: O código possui como objetivo realizar a leitura de dois arquivos JSON, broken_database_1 e broken_database_2 corrompidos e recuperar os dados modificados, realizando a troca de caracteres que foram alterados e a mudança do tipo de dados numéricos que estavam como tipo string para o tipo number. Após isso, o código deve exportar os arquivos corrigidos e realizar a inserção no banco de dados da plataforma SQL online. Por meio das informações obtidas deve-se criar um relatório de vendas que responda às perguntas solicitadas.

Definições: Foram definidos os dois arquivos iniciais corrompidos e logo após a chamada das funções que realizam as modificações. Para realizar a construção do código foi utilizado a instrução "**requeire('fs')**" que permite utilizar o módulo fs do **nodejs**.

```
// Chamada das funções com os arquivos originais

const repairedData1 = fixData ('broken_database_1.json');
const repairedData2 = fixData ('broken_database_2.json');

// Chamada das funções que exporta os arquivos corrigidos

exportFiles('fixed_database_1.json', repairedData1);
exportFiles('fixed_database_2.json', repairedData2);
```

Funções:

changCaracters: A função possui o objetivo de varrer os nomes dos veículos e marcas alterando os caracteres corrompidos "æ" para o carácter "a" e o carácter "ø" para o carácter "o". Para realizar essa correção o seguinte código foi utilizado:

```
function changCaracters (data) {
    if (typeof data === 'string') {
        data = data.replace(/æ/g, 'a').replace(/ø/g, 'o');
        return data;
    } else if (typeof data === 'object') {
    for (let key in data) {
        data[key] = changCaracters(data[key]);
    }
}
    return data;
}
```

A função recebe como parâmetro de entrada a variável data, que contém o arquivo a ser analisado. Por meio do operador “**typeof”** a função verifica se o data é do tipo string, sendo ela do tipo string, a função procura e substitui as ocorrências por meio do método “**replace**” e em seguida retorna a string modificada. Caso a variável data seja do tipo objeto, a função percorre todas as propriedades do objeto aplicando as mesmas substituições nos caracteres, realizando esse procedimento até que todos os valores encontrados sejam do tipo string. No final a função retorna o objeto data com as modificações realizadas.

changType: A função possui o objetivo de varrer os valores das vendas alterando os valores corrompidos do tipo string para o tipo number. Para realizar essa correção o seguinte código foi utilizado:

```
function changType(data) {
    if (typeof data === 'object') {
        for (let key in data) {
            if (key === 'vendas') {
                data[key] = Number(data[key]);
            } else {
                data[key] = changType(data[key]);
            }
        }
    }
return data;
}
```

A função recebe como parâmetro de entrada a variável data, que contém o arquivo a ser analisado. A função primeiramente verifica se o parâmetro data é do tipo objeto, caso seja, por meio do **for** ele varre os pares de chave em todo o objeto. Após isso, a função verifica se entre o par de chaves há o nome “vendas”. Caso seja

encontrado o valor correspondido é convertido para número, utilizando a função "**Number**" e assim armazenada no objeto, caso não seja, a função é chamada recursivamente, passando o valor atual como parâmetro. No final a função retorna o objeto data com as modificações realizadas.

fixData: A função possui o objetivo de ler o conteúdo do arquivo analisado como JSON e realizar a chamada das funções de reparo. Para realizar essa tarefa o seguinte código foi utilizado:

```
function fixData(filePath) {

    const data = JSON.parse(fs.readFileSync(filePath, 'utf8'));
    const dataFixed = changCaracters(data);
    const dataFixedsales = changType(dataFixed);
    return dataFixedsales;
}
```

A função recebe como parâmetro de entrada os arquivos **broken_database_1** e **broken_database_2**, lidos por meio do **filePath**, ela cria a variável data e atribuindo para ela a leitura do arquivo, interpretando-o como um arquivo JSON. Após isso a função cria e atribui as várias, **dataFixed** e **dataFixedsales** as funções changCaracters e changType. A função changCaracters recebe o arquivo data como parâmetro de entrada e a função **changType** recebe como parâmetro de entrada a saída da função anterior. No final a função torna a variável **dataFixedsales** com o arquivo modificado.

exportFiles: A função possui o objetivo de reescrever e exportar o arquivo JSON corrigido. Para realizar essa tarefa o seguinte código foi utilizado:

```
function exportFiles(filePath, data) {
    const jsonData = JSON.stringify(data);
    fs.writeFileSync(filePath, jsonData, 'utf8');
}
```

A função recebe como parâmetro de entrada o **filePath** que realiza a leitura das saídas fixed_dabase_1 e fixed_database_2 e o objeto data, que contém os dados corrigidos a serem exportados. A função cria a variável **jsonData** e a define como sendo do tipo JSON referente aos dados fornecidos utilizando o método **JSON.stringify**, logo após por meio do método **fs.writeFileSync** os dados são escritos no arquivo especificado pelo filePath.