media
.monks

Projeto Classificatório

Processo seletivo – Tech

João Luiz da Costa Dias

2024

media
.monks

Sumário

media
.monks

Introdução

Projeto destinado ao processo seletivo da Media.Monks, tendo como objetivo elaborar uma solução para o problema de recuperação de dados originais de dois bancos de dados corrompidos e a elaboração de um relatório de vendas baseado nos banco corrigidos. Para a elaboração da solução foi utilizado a linguagem JavaScrpit para a correção e para o relatório a linguagem SQL, como descrito nos requisitos do projeto.

Código JavaScrpit

O código JavaScript realiza a leitura das bases de dados "broken_database_1" e "broken_database_2", além de efetuar a manipulação para corrigir problemas relacionados a caracteres corrompidos e a tipos de dados incorretos, transformando strings em números conforme necessário. Posteriormente, exporta dois arquivos já corrigidos.

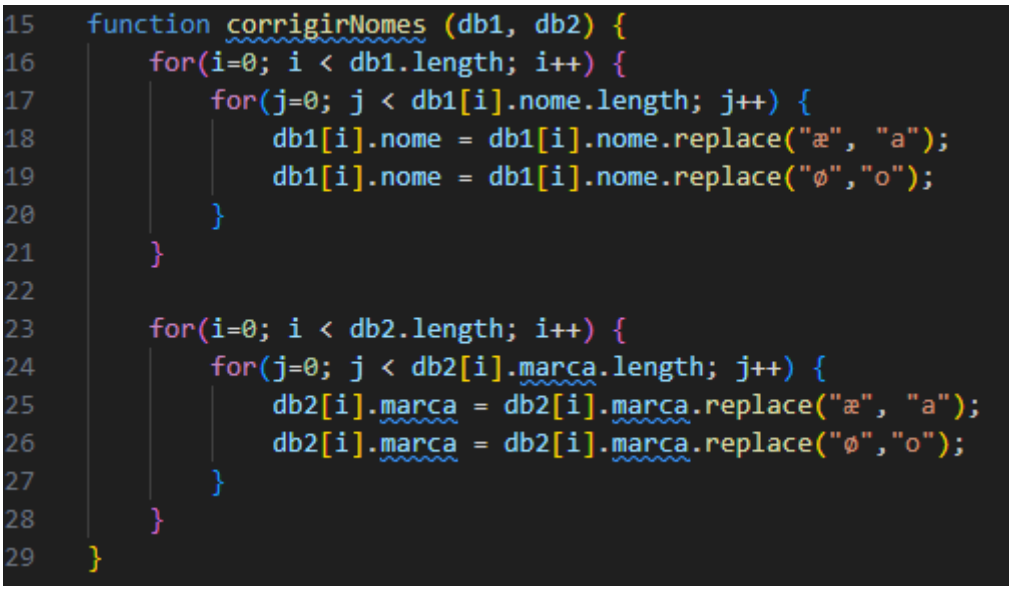
A primeira função lerJSONs() descritana Figura 1, realiza a leitura dos dois arquivos utilizando a função "require()" e armazena os dados em duas variáveis, denominadas "dataBase1" e "dataBase2". Em caso de algum erro, a estrutura de bloco "try-catch" valida se a execução foi bem-sucedida, exibindo mensagens no console para indicar se o código foi executado corretamente ou se ocorreu algum erro.

Figura 1- Função lerJSONs

```
3   function lerJSONs() {
4       try {
5           dataBase1 = require ("./json's_originais/broken_database_1.json");
6           dataBase2 = require ("./json's_originais/broken_database_2.json");
7           console.log("Deu certo");
8       } catch(error) {
9           console.log("Erro ao ler os arquivos");
10      }
11  }
12
13  lerJSONs();
```

A função corrigirNomes(db1, db2) descrita na Figura 2 realiza a manipulação dos dados contidos nas bases de dados, corrigindo os nomes das marcas e dos veículos, e o do tipo de dado das vendas. Ela percorre os objetos db1 e db2, substituindo os caracteres "ø" e "æ" pelos caracteres "o" e "a", respectivamente, nos campos de nome e marca dos objetos. A função corrigirVendas converte o valor do campo "vendas" do tipo string para o tipo number, utilizando a função Number(), como descrito na Figura 3.

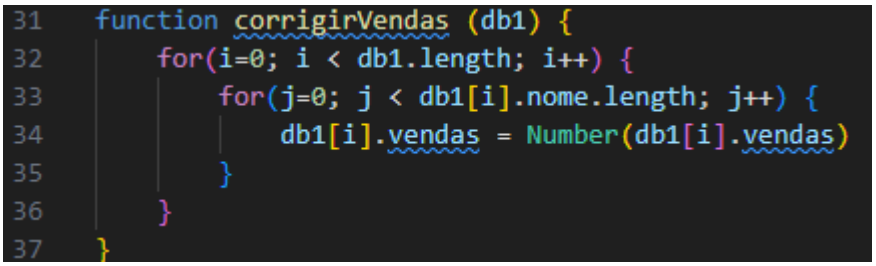
Figura 2- Função corrigirNomes



```
function corrigirNomes (db1, db2) {
    for(i=0; i < db1.length; i++) {
        for(j=0; j < db1[i].nome.length; j++) {
            db1[i].nome = db1[i].nome.replace("æ", "a");
            db1[i].nome = db1[i].nome.replace("ø","o");
        }
    }

    for(i=0; i < db2.length; i++) {
        for(j=0; j < db2[i].marca.length; j++) {
            db2[i].marca = db2[i].marca.replace("æ", "a");
            db2[i].marca = db2[i].marca.replace("ø","o");
        }
    }
}
```

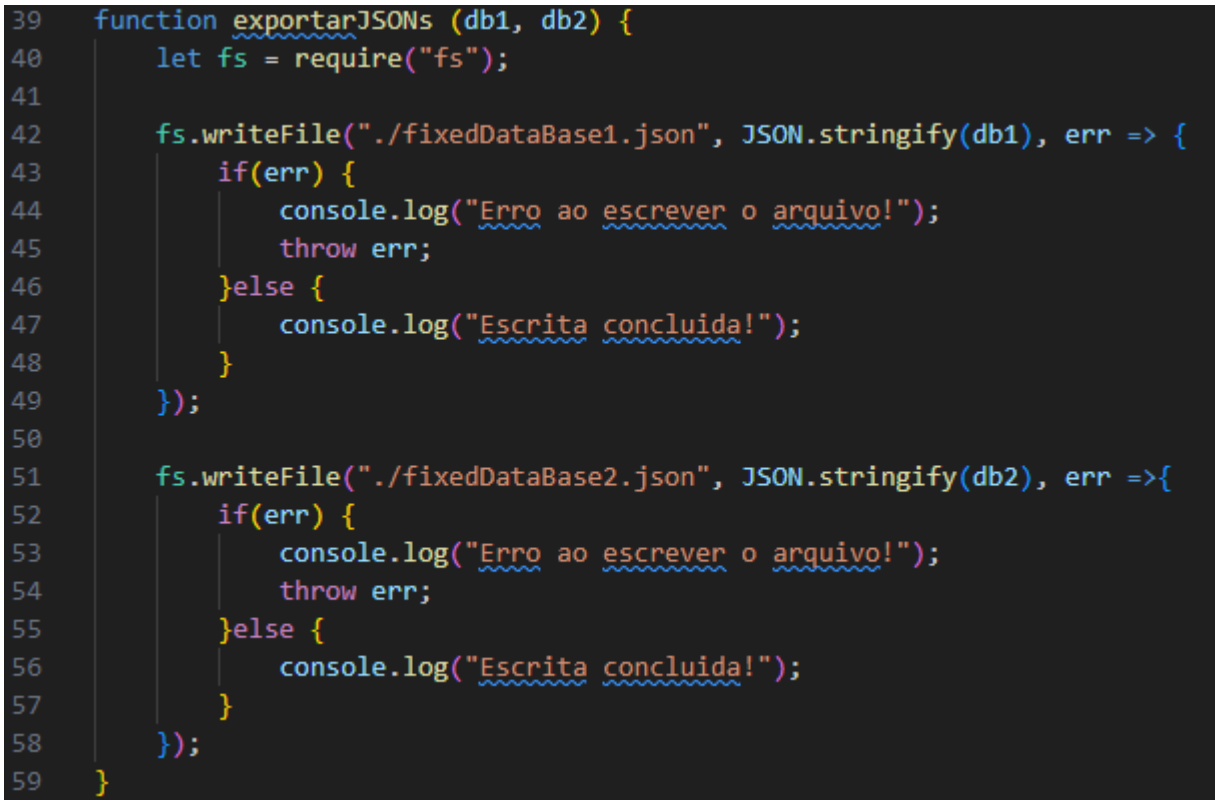
Figura 3- Função corrigirVendas



```
function corrigirVendas (db1) {
    for(i=0; i < db1.length; i++) {
        for(j=0; j < db1[i].nome.length; j++) {
            db1[i].vendas = Number(db1[i].vendas)
        }
    }
}
```

A função exportarJSONs(db1, db2) descrita na Figura 4 é responsável por exportar os dados corrigidos para novos arquivos JSON. Ela utiliza o módulo fs do node.js para gravar dados em um arquivo JSON e a função writeFile para escrever os dados nas novas bases de dados. Caso ocorra algum erro na escrita, mensagens de erro são exibidas no console.

Figura 4- Função exportarJSONs



```
function exportarJSONs (db1, db2) {
    let fs = require("fs");

    fs.writeFile("./fixedDataBase1.json", JSON.stringify(db1), err => {
        if(err) {
            console.log("Erro ao escrever o arquivo!");
            throw err;
        }else {
            console.log("Escrita concluida!");
        }
    });

    fs.writeFile("./fixedDataBase2.json", JSON.stringify(db2), err =>{
        if(err) {
            console.log("Erro ao escrever o arquivo!");
            throw err;
        }else {
            console.log("Escrita concluida!");
        }
    });
}
```

Código SQL

O código SQL cria a tabela chamada tabelao. O comando CREATE TABLE define a estrutura da tabela, com seis colunas: data, id_marca, marca, nome_veiculo, vendas, valor_veiculo. O comando INSERT INTO especifica a tabela de destino (tabelao) e as colunas nas quais os dados serão inseridos. Em seguida, o comando SELECT é utilizado para definir as informações que serão inseridas, tomando como base os arquivos corrigidos e direcionando as suas colunas. O comando INNER JOIN é utilizada para relacionar as duas tabelas pela coluna c2 da tabela database_1 e pela coluna c1 da tabela database_2, sendo as duas o id_marca o que possibilita o relacionamento por uma coluna em comum.