rocky
.monks

Projeto Classificatório

Processo seletivo - TI

Bianca Vitorino de Souza Melaré

2023

Sumário

Código Java Script

O código em JavaScript realiza a leitura, manipulação e exportação de dados contidos em dois arquivos JSON: broken_database_1.json e broken_database_2.json.

A função lerJson() (Figura 1) é responsável por ler os dados desses arquivos utilizando a função require() e armazená-los nas variáveis database_1 e database_2. Para realizar o tratamento de erro foi utilizado a estrura “try/catch”, em que se ocorrer algum erro será exibido no console a mensagem “Erro ao ler o arquivo.” e na linha seguinte o erro que foi gerado.

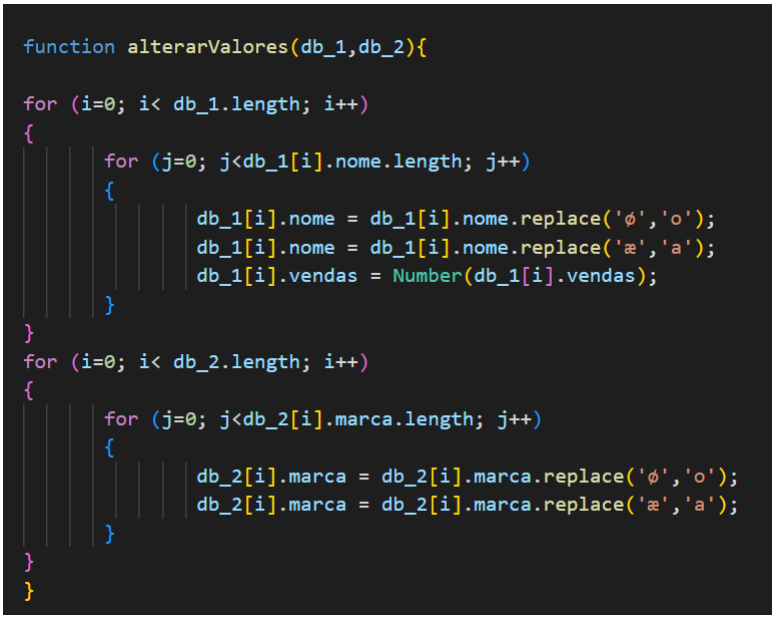
Figura 1: Função lerJson()

```
var database_1,database_2;

function lerJson(){
  try{
    database_1 = require("./json/broken_database_1.json");
    database_2 = require("./json/broken_database_2.json");
  } catch(error){
    console.log("Erro ao ler o arquivo." + "\n" + error);
  }
}
```

Fonte: Autoria própria

A função alterarValores(db_1,db_2) (Figura 2) realiza a manipulação dos dados contidos nas bases de dados, corrigindo os nomes das marcas e dos veículos, além do tipo de dado das vendas. Ela percorre os objetos db_1 e db_2, substituindo os caracteres "ø" e "æ" pelos caracteres "o" e "a", respectivamente, nos campos de nome e marca dos objetos. Ademais, ela converte o valor do campo "vendas" do tipo string para o tipo number, utilizando a função Number().

Figura 2: Função alterarValores(db_1,db_2)

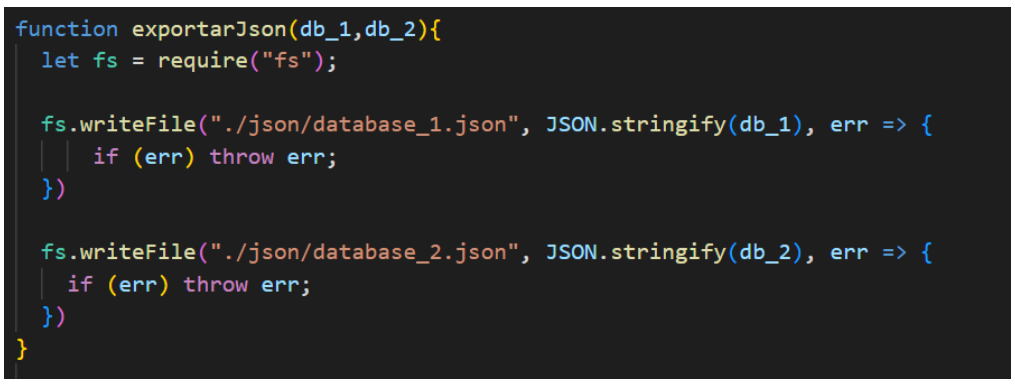


```
function alterarValores(db_1,db_2){

for (i=0; i< db_1.length; i++)
{
        for (j=0; j<db_1[i].nome.length; j++)
        {
            db_1[i].nome = db_1[i].nome.replace('ø','o');
            db_1[i].nome = db_1[i].nome.replace('æ','a');
            db_1[i].vendas = Number(db_1[i].vendas);
        }
}
for (i=0; i< db_2.length; i++)
{
        for (j=0; j<db_2[i].marca.length; j++)
        {
            db_2[i].marca = db_2[i].marca.replace('ø','o');
            db_2[i].marca = db_2[i].marca.replace('æ','a');
        }
}
}
```

Fonte: Autoria própria

Por fim, a função exportarJson(db_1,db_2) (Figura 3) é responsável por exportar os dados manipulados 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 database_1.json e database_2.json. Se ocorrer algum erro na escrita, a mensagem de erro será exibida no console.

Figura 3: Função exportarJson(db_1,db_2)

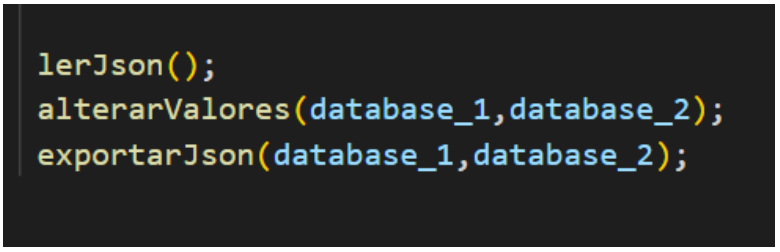


```
function exportarJson(db_1,db_2){
  let fs = require("fs");

  fs.writeFile("./json/database_1.json", JSON.stringify(db_1), err => {
      if (err) throw err;
  })

  fs.writeFile("./json/database_2.json", JSON.stringify(db_2), err => {
    if (err) throw err;
  })
}
```

Fonte: Autoria própria

As chamadas das três funções (lerJson(), alterarValores(database_1,database_2) e exportarJson(database_1,database_2)) são realizadas no final do código (Figura 4).

Figura 4: Chamada das funções



Fonte: Autoria própria

Código SQL

O código SQL cria uma nova tabela chamada tabela_unificada.

A cláusula CREATE TABLE define a estrutura da nova tabela, com seis colunas: id_marca, marca, data, vendas, valor_veiculo e nome_veiculo.

A cláusula INSERT INTO especifica a tabela de destino (tabela_unificada) e as colunas nas quais os dados serão inseridos. Em seguida, a cláusula SELECT é usada para definir as informações que serão inseridas.

Nessa cláusula SELECT, são selecionados seis colunas de duas tabelas diferentes: c1, c2, c3, c4, c5 da tabela database_1 e c1 e c2 da tabela database_2. A cláusula INNER JOIN é utilizada para relacionar as duas tabelas pela coluna c2 da tabela database_1 e pela coluna c1 da tabela database_2.

Assim, cada linha da tabela tabela_unificada conterá informações de duas tabelas diferentes, relacionadas por uma coluna em comum.