Correção de banco de dados

1. Introdução

Esta documentação visa explicar o funcionamento da solução proposta no arquivo resolucao.js.

A explicação das funções foi dividida em quatro seções:

Manipulação de arquivos .json: leitura e escrita do banco de dados em formato JSON no sistema de arquivos do computador.

Correção dos dados: manipulação dos dados em si, visando corrigir três problemas: caracteres trocados, preço salvo como string em vez de number e quantidade ausente nos produtos fora de estoque.

Validação: teste das soluções implementadas através de duas funções: ordenação dos dados e soma do valor do estoque por categoria.

Execução do programa: função que controla a ordem de execução de todas as outras.

É importante salientar que, para a execução correta do programa, **o arquivo bronken-database.json precisa estar na mesma pasta que o script resolucao.js**.

2. Explicação das funções

a. Manipulação de arquivos .json

readJSON() - responsável por ler o arquivo broken-database.json no sistema de arquivos e convertê-lo para um objeto salvo na variável database.

Foi utilizado o módulo fs presente no node.js para ler o arquivo através do método readFileSync. A versão síncrona foi utilizada para que o restante do código não seja executado sem o banco de dados presente.

O método entrega o arquivo JSON em uma string, então foi utilizado JSON.parse para convertê-lo em um objeto.

writeJSON() - converte o banco de dados corrigido para uma string utilizando o método JSON.stringify. A string é então escrita no arquivo saida.json utilizando o

método writeFileSync presente no módulo fs. A conversão para string é necessária para que writeFileSync aceite o argumento.

b. Correção dos dados

fixName() - corrige os caracteres trocados. Itera sobre o array database e, em cada objeto, procura a propriedade name.

Ao encontrar a propriedade name, é utilizado o método replace para encontrar o caractere æ e substituir por ‘a’. O mesmo é feito com os caracteres ¢, ø e ß, substituídos respectivamente por ‘c’, ‘o’, e ‘b’.

Em todas as chamadas ao método replace foi utilizada a flag /g, para que sejam substituídas todas as ocorrências do caractere em questão, e não apenas a primeira.

fixPrice() - Itera sobre o array e, para cada objeto, encontra a propriedade price e utiliza o método parseFloat para atribuir um novo valor à propriedade, com o tipo number.

Foi utilizado parseFloat em vez de parseInt para que as casas decimais fossem preservadas.

fixQuantity() - Itera sobre o array e, para cada objeto, procura a propriedade quantity. Caso ela não exista, cria a propriedade quantity com o valor 0.

c. Validação

sortJSON() - ordena os dados do banco de dados considerando dois fatores, primeiro a categoria, e depois o ID de cada produto. Utiliza o método sort com uma função que compara a propriedade category de cada objeto. Se os dois tiverem o mesmo valor, é retornado o resultado do id de a menos o id de b.

Caso os objetos tenham a propriedade category com valores diferentes, é retornado 1 caso a categoria de a seja superior a de b, ou -1 em caso contrário.

Ao final, é utilizado o método console.table para exibir o banco de dados ordenado.

sumPrice() - Soma o valor total do estoque de cada categoria, considerando a quantidade de cada produto. Primeiramente é criado um objeto vazio, chamado sum, para armazenar as somas.

Iterando sobre o array, para cada objeto é obtido o valor da propriedade category e realizado um teste condicional:

Caso a propriedade não exista no objeto sum, ela é criada com o valor do preço do item sendo analisado multiplicado pela sua quantidade.

Caso a propriedade já exista no objeto sum, seu valor é acrescido do preço do item sendo analisado multiplicado pela sua quantidade.

Novamente é utilizado o método console.table para exibir o resultado.

d. Execução do programa

run() - função simples apenas para executar todas as funções anteriores na ordem correta ao inicar o script com o comando node.