Gabriel Augusto da Silva Lima

Relatório referente a resolução do arquivo broken-database.Json

O algoritmo do projeto será divido em dois grandes tópicos explicados dentro do arquivo através dos comentários.

Primero teremos o tópico de Leitura do banco de dados e suas correções sendo aplicadas.

Dentro do primeiro tópico teremos uma função principal e outras três subfunções responsáveis pelas correções, essas funções são:

1 – lerBancoDados() => Essa função será responsável por armazenar tudo que envolver a leitura do banco com suas correções a partir de uma variável que contém o banco de dados incorreto. Então dentro delas teremos um:

1.1. **“For”** – Será o loop responsável por caminhar em todos os elementos do banco de dados, loop esse que será primordial para aplicarmos as correções.

1.2. **Função lerNomes ()** – Essa função irá receber apenas as propriedades “name” que serão lidas com o “for” a partir do banco de dados.

1.2.1. Ela irá receber os valores errados a partir da variável: “receberDataBaseNames” e será aplicado um “replace()” para substituir os caracteres errados pelos corretos e vinculá-los a variável “caracteresCorretos” que serão devolvidos ao objeto: broken-database. Com isso, se caso futuros nomes sejam aplicados ao banco de dados, a função irá ler o nome e se caso houver algum erro no caractere, irá corrigi-lo.

1.3. **Função lerQuantidades()** => A partir do “for” essa função irá receber todos as propriedades do tipo “quantity” que estão no objeto broken-database e aplicara a variável “receberDataBaseQuantidade.

1.3.1. A variável entrará numa estrutura de controle “if” que irá aplicar o seguinte controle: Se caso tenha alguma propriedade “quantity” no banco de dados que retorne “undefined”, ou seja, que não esteja definida, faremos com que seja adicionado o valor zero a essa propriedade. E ao fim o valor é retornado ao banco de dados

1.3.2. Com isso, se no futuro adicionarmos mais algum objeto que tenha a propriedade “quantity”, se caso ela não tenha seu valor definido, será igualado a 0.

1.4. **Função lerPrecos()** => Essa função será responsável por receber a partir do “for” todas as propriedades “price” do objeto broken-database e armazenar na variável “receberQuantidadePreco”, iremos aplicar a essa variável um “parseFloat()” e fazer com que todos os valores que entrem nessa propriedade, sejam alterados para esse tipo de dado. Aplicamos o valor após o parseFloat a variável “dataBasePrecosCorretos” e a retornamos ao objeto broken-database.

Por fim chamamos a execução de todas as funções e partimos ao segundo tópico onde temos as funções de validação.

No segundo tópico temos duas funções que irão servir para corrigir o banco de dados. Essas funções são:

1. somaCategoria() => Essa função irá receber a função “lerBancoDados()” que já contém o objeto broken-database corrigido e aplicar um “filter()” com isso iremos dividi-lo pelas suas respectivas categorias e atribuir a quatro variáveis: “arrayEletronicos, arrayAcessorios, arrayEletrodomesticos e arrayPanelas”, retornando a partir de um “reduce()” a soma das quantidades que categoria contém.

2. ordenar() => Essa função irá receber o objeto broken-database corrigido e será responsável por aplicar um “filter()” a partir de cada categoria e as atribuí-las as suas respectivas variáveis, sendo elas: “acessorios”, “eletronicos”, “eletrodomesticos” e “panelas”, e os ordenar por ordem crescente da propriedade “id” usando o método “sort()” .

   Obs: Na função “ordenar()” não consegui aplicar de forma exata o que foi pedido, não encontrei uma forma de aplicar o método sort() mais uma função de comparação que ordene o objeto por ordem Alfabética e ordem crescente de Id na mesma saída.

3. Por último foi criado a função exportarSaidaJson() que será responsável por criar um novo arquivo chamado “saída.json” e criar um novo objeto com o broken-database corrigido.