Teste Prático – TI

ROCKY

João Luiz Fernandes

Sumário

Javascript

Entre as razões que me levaram a escolher o javascript para a implementação da solução do desafio, está, certamente, a minha maior experiência com a linguagem. Sua presença em ambas as frentes do desenvolvimento me permitiu resgatar a experiência com os mais variados projetos anteriores. Por fim, soma-se isso a node.js e noSQL *databases* me parecem ser uma combinação bem popular, e a linguagem estava decidida.

Tratamentos de bugs

Na importação dos dados do JSON, optei por utilizar uma declaração *try...catch* para o tratamento de bugs. Caso o processo seja mal sucedido, o que pode ocorrer, por exemplo, quando o arquivo de origem dos dados 'broken-database' não estiver presente no diretório, *catch* recebe o erro e o apresenta no terminal.

Porém, na próxima vez que o tratamento de bugs se mostrou necessário, na exportação de dados, decidi por utilizar a própria *callback* da função que a realiza, a fim de demonstrar pluralidade de métodos de implementação. A função 'fs.writeFile' recebe os seguintes parâmetros: o alvo da exportação, o conteúdo da exportação e uma função que recebe um erro e, caso ele não seja nulo, o imprime no terminal.

Funcionabilidades do código

Funções de importação e exportação de dados

Para a interação com o arquivo JSON fornecido, é primeiramente importado a API File System.

getData

Através de sua função da ‘existsSync’, é realizado a validação do arquivo JSON, e, caso bem sucedida, os dados são importados pela ‘fs.readFileSync’ e convertidos para uma matriz de objetos, para que, por fim, possam ser retornados à inicialização da variável ‘brokenInput’.

saveData

Os dados da variável ‘brokenInput’ são convertidos para JSON, para que, então, a função ‘fs.writeFile’ os salve no arquivo ‘saida.json’, estando esse a ser criado caso não exista.

Funções de recuperação dos dados originais

fixNames

Modifica o nome de todos os produtos contidos na matriz de ‘brokenInput’ para o resultado da função ‘replaceCorruptChars’. Esta, por sua vez, atua no relacionamento de cada letra do nome do produto a elementos de uma matriz que apresenta as possíveis formas de um caractere corrompido (charState.corrupt), e na eventual troca dessas letras para seu estado original, por meio de uma correspondência de índices com a matriz ‘charState.original’.

fixPrices

Percorre o preço dos produtos e os reatribui como números.

fixQuantitites

Percorre a quantidade dos produtos e, caso ela esteja definida, atribui o valor de 0.

Funções de validação

printProductList

Cria uma cópia da matriz de produtos contida em ‘brokenInput’, os ordena sob os critérios de categoria em ordem alfabética e id em ordem crescente, e imprime seus nomes.

printInvetoryValue

Com o método funcional *reduce*, a função constrói um objeto que contém o valor total do estoque de cada categoria, e, em seguida, os imprimi no terminal, seguido do valor total de estoque.