

Teste prático Rocky

Natham Abdala Merlin Coracini

Sumário

1	Funcionalidades	1
1.1	Ler o arquivo JSON	1
1.2	Corrigir nomes	1
1.3	Corrigir preços	1
1.4	Corrigir quantidades	1
1.5	Exportar um arquivo JSON com o banco corrigido	1
2	Escolha da linguagem	2
3	Tratamentos no código	2
4	Outros	2

1 Funcionalidades

A seção a seguir aborda as diferentes funcionalidades do programa feito.

1.1 Ler o arquivo JSON

Essa função trata a leitura do arquivo JSON (cujo nome é fornecido no parâmetro 'filename'), na qual é feita através do método `readFile` e de forma assíncrona, utilizando `async await`. Em seguida, o arquivo é convertido de uma `String` para um objeto Javascript (através da função `JSON.parse()`), que é o formato na qual a maior parte do programa irá processar. Por fim, o array de objetos Javascript é retornado.

1.2 Corrigir nomes

Essa função primeiramente converte o banco de dados para o formato de `String`, para que seja mais fácil encontrar os caracteres a serem corrigidos (æ, ç, ø, ð). Em seguida, é usada a função `String.prototype.replace()` para encontrar (através de `Regex`) todas as ocorrências desses caracteres e substituí-los pelo valor correto. Finalmente, o array resultante dessa operação é convertido de volta para um objeto Javascript e é atribuído como retorno da função. Foi necessário criar um novo Array pois não é possível modificar os caracteres de uma `String` em Javascript.

1.3 Corrigir preços

Essa função verifica o tipo de todos os valores do atributo `price` e, caso seja `String`, converte através de `parseInt` para um inteiro e substitui esse valor no banco de dados.

1.4 Corrigir quantidades

Essa função verifica no banco de dados todos os objetos que não possuem o atributo `quantity`, e, caso não possuam, cria esse campo com o valor 0.

1.5 Exportar um arquivo JSON com o banco corrigido

Essa função exporta o banco de dados através de `writeFile` de forma síncrona para um arquivo de nome 'filename' (parâmetro da função)

2 Escolha da linguagem

A escolha de Javascript foi muito positiva para o desenvolvimento desse teste, pois facilitou diversas operações que em outras linguagens demandaria a implementação manual. Como exemplo dessas operações temos o tratamento de erros, a função `String.prototype.replace()`, a indexação pelo nome de um atributo, a leitura de um arquivo JSON e a conversão desse para um objeto Javascript.

3 Tratamentos no código

O tratamento de erros no código foi feito através de blocos `try catch`, pela checagem de nulo e pelo método `isNaN()`. Os métodos de leitura e escrita de arquivo foram cercados por blocos `try catch` para garantir, respectivamente, que o arquivo a ser lido existe e que há espaço suficiente no disco para escrever o arquivo, entre outros erros. O método `JSON.parse()` também foi tratado para garantir que o banco de dados é um arquivo JSON válido. No caso do método `parseInt`, foi realizada uma verificação `isNaN` para garantir que a String realmente pode ser convertida para um inteiro, e portanto armazenar somente valores válidos no banco de dados de saída.

4 Outros

Em geral, achei tranquilo o desenvolvimento do projeto. Não possuía muito conhecimento dos métodos prontos de Javascript para substituir caracteres em uma string (`String.replace`) e também não conhecia o funcionamento de Regex, mas estudei um pouco de seu funcionamento e consegui aplicar no trabalho.