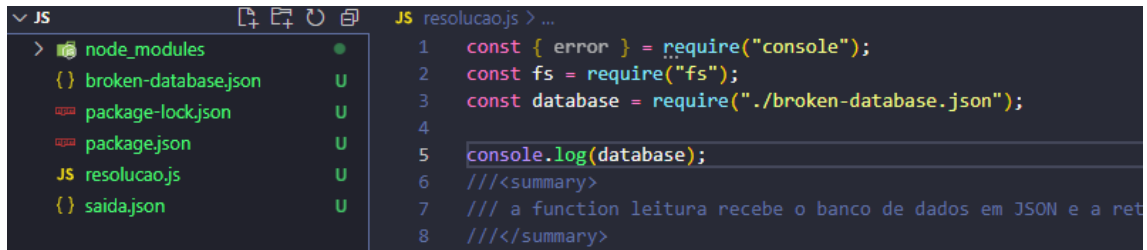


Projeto resolução da Rocky

1 Início do código

Primeiro foi necessário preparar o código e o visual Studio code para usar o JSON e o Node. Foi baixado o Node e depois iniciado dentro do projeto.



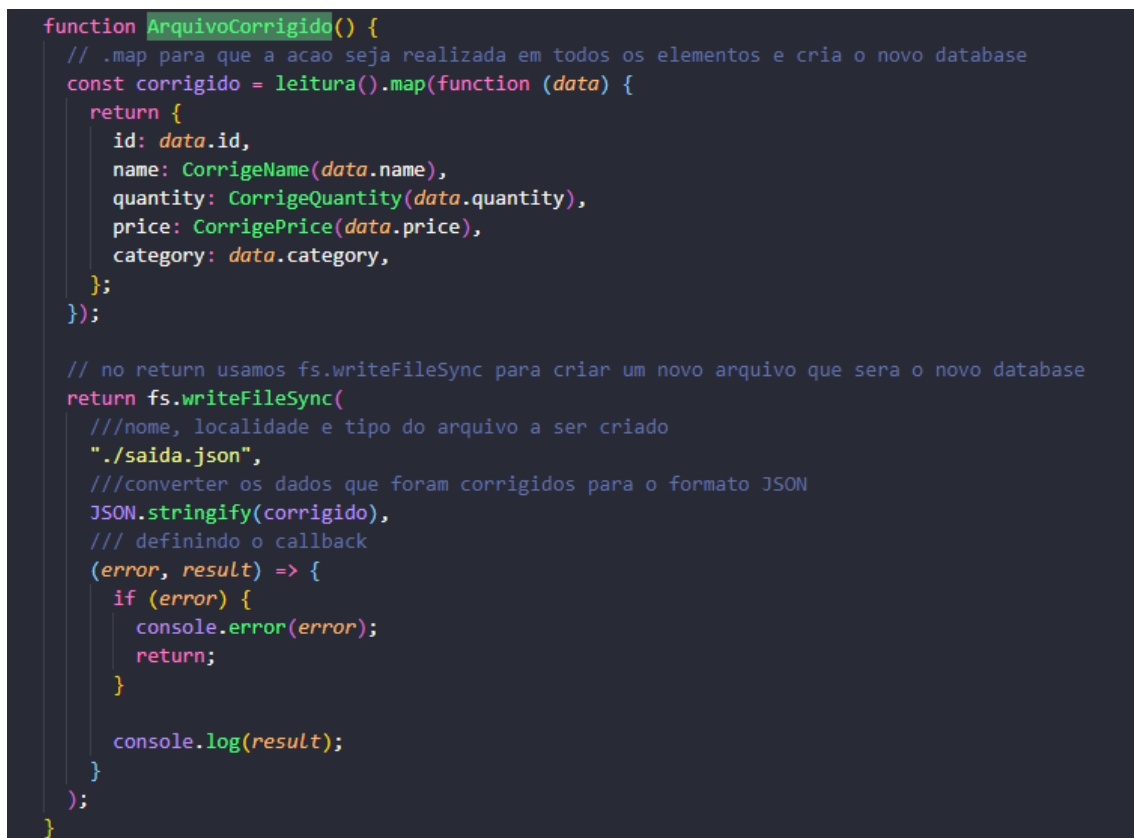
```
JS resolucao.js > ...
1  const { error } = require("console");
2  const fs = require("fs");
3  const database = require("./broken-database.json");
4
5  console.log(database);
6  ///<summary>
7  /// a function leitura recebe o banco de dados em JSON e a ret
8  ///</summary>
```

Para usar o JSON no código foi usado os require. Para chamar o banco de dados e os comandos fs.

2. Funções

ArquivoCorrigido()

Essa função é principal de todo código, ela responsável por gerar um novo banco de dados, gerando o arquivo saida.json, corrigindo os erros do antigo. Ela compila outras quatro funções: leitura(), CorrigeName(), CorrigePrice(), CorrigeQuantity.



```
function ArquivoCorrigido() {
  // .map para que a acao seja realizada em todos os elementos e cria o novo database
  const corrigido = leitura().map(function (data) {
    return {
      id: data.id,
      name: CorrigeName(data.name),
      quantity: CorrigeQuantity(data.quantity),
      price: CorrigePrice(data.price),
      category: data.category,
    };
  });

  // no return usamos fs.writeFileSync para criar um novo arquivo que sera o novo database
  return fs.writeFileSync(
    ///nome, localidade e tipo do arquivo a ser criado
    "./saida.json",
    ///converter os dados que foram corrigidos para o formato JSON
    JSON.stringify(corrigido),
    /// definindo o callback
    (error, result) => {
      if (error) {
        console.error(error);
        return;
      }

      console.log(result);
    }
  );
}
```

A função funciona da seguinte maneira, ao ser criada a variável constante `corrigido`, será alocado nela a função `leitura()` junto ao método `map`, que em seus parâmetros e criada uma função, com parâmetro `data`, a função de `map` retornara os valores do objetos usando as outras três funções para corrigi-los.

Por fim a própria função `ArquivoCorrige()` retorna o comando `fs.writeFileSync`, para criar um novo arquivo, de nome `saida.json`, os dados alterados que estavam alocados na variável constante `corrigido` são convertidos para o formato JSON, pelo comando `JSON.stringify()`, logo em seguida um callback para casos de erros e por último um `console.log` exibindo o resultado.

Leitura()

Primeira função do código ele é importante porque ela é que nos permite usar os dados contidos no Broken-database.

```
///<summary>
/// a function leitura recebe o banco de dados em JSON e a retorna em um formato para ser usado.
///</summary>
function leitura() {
  //fs.readFileSync le o banco de dados em formato JSON, de nome broken-database.
  // e armazena na variavel na variavel recebe_JSON.
  const recebe_JSON = fs.readFileSync("./broken-database.json");
  // atraves do JSON.parse ele converte para o formato javascript.
  // e armazena na variavel na variavel dados.
  const dados = JSON.parse(recebe_JSON);
  return dados;
}
```

Para que podemos os dados do banco ela converte os dados no formato JSON em javascript.

CorrigeName()

Um dos maiores problemas no broken-database é o fato dos caracteres no valor nome estarem errados.

```
///<summary>
/// a function CorrigirNome utiliza o replace para alterar os caracteres que apresentavam erro.
///</summary>
function CorrigirNome(name) {
  // define quais caracteres serao mudados e para quais.
  const correcao = { æ: "a", "ç": "c", ß: "b", ø: "o" };
  // /gi: g = global i=ignore
  return name.replace(/æ|ç|ß|ø/gi, function (retorno) {
    return correcao[retorno];
  });
}
```

Para resolver isso foi necessário criar uma variável constante que nos atributos coloquei os caracteres defeituosos e coloquei o valor que será trocado. Em seguida vem o `return` nele é aplicado o parâmetro da função junto ao método `replace` e o `gi`, para que a função seja executada em todo o array.

CorrigeQuantity()

Para corrigir problema de que alguns elementos não possuem o valor quantity, quando o mesmo era igual a 0. Foi criada essa função.

```
///<summary>
/// corrige o valor de quantidade que avia sumido.
///</summary>
function CorrigeQuantity(quantity) {
    // se o objeto tiver um valor ele mantem mas se nao ele adiciona o valor 0
    if (quantity > 0) return quantity;
    else return 0;
}
```

O if executa a seguinte condição, se o valor de quantity for maior que zero, ele retorna o quantity, ou seja, o valor é esse mesmo por isso se matem do jeito que está, porém se ele for menor que zero, no caso de ser zero ou não haver valor ele retorna o valor 0, atribuindo esse valor o 0.

CorrigePrice()

Para corrigir o valor price que deveria ser no formato numérico, mas está no formato caractere.

```
///<summary>
/// retorna o preco no formato correto
///</summary>
function CorrigePrice(price) {
    // usa o parseFloat para transformar do formato string para float.
    return parseFloat(price);
}
```

No return usa o método `parseFloat()` para converter o valor de string para float.

OrdenaSaida()

Depois da criação do novo banco de dados, agora com seus dados corrigidos, será feita a organização dos objetos por categoria e em seguida por id.

```

///<summary>
/// A OrdenaSaida vai mostrar objetos na ordem estipulada
///</summary>
function OrdenaSaida() {
  // recebe o novo database
  const recebe_JSON = fs.readFileSync("saida.json");
  const dados = JSON.parse(recebe_JSON);

  // o sort para ordenar segundo os parametros
  dados.sort(function (a, b) {
    if (a.category < b.category) {
      return -1;
    } else {
      if (a.category == b.category && a.id < b.id) {
        return -1;
      }
    }
  });
  return console.log(dados);
}

```

Primeiro recebemos o novo banco de dados na variável `dados`, em seguida usando o método `sort` se cria um função, com parâmetros `a` e `b`, com `if` criamos a condição se `a.category` menor que `b.category` então `return -1`, em seguida um `else` com a condição se `a.category` for igual `b.category` e `a.id` menor `b.id` então `return -1`. No final da função então retorne `dados`.

Estoque()

Primeiro recebemos o novo banco de dados na variável `dados`, em seguida criamos novas variáveis para receber os valores a serem calculados, elas tem que ser do tipo `var`, ou seja não constante, pois os valores serão mudados

constantemente.

```
function Estoque() {
  var recebe_JSON = fs.readFileSync("saida.json");
  var dados = JSON.parse(recebe_JSON);
  var panelas = 0;
  var Eletrodomesticos = 0;
  var Eletronicos = 0;
  var Acessorios = 0;

  // o forEach para executar a funcao do calculo em todos elementos selecionados
  dados.forEach((data) => {
    if (data.category == "Panelas") {
      panelas = panelas + data.quantity * data.price;
    }

    if (data.category == "Eletrodomésticos") {
      Eletrodomesticos = Eletrodomesticos + data.quantity * data.price;
    }

    if (data.category == "Eletrônicos") {
      Eletronicos = Eletronicos + data.quantity * data.price;
    }
    if (data.category == "Acessórios") {
      Acessorios = Acessorios + data.quantity * data.price;
    }
  });
  console.log("Valor total de panelas:", panelas);
  console.log("Valor total de Eletrodomesticos:", Eletrodomesticos);
  console.log("Valor total de Eletronicos:", Eletronicos);
  console.log("Valor total de Acessorios:", Acessorios);
}
```

Com a variável `dados` usamos o método `forEach`, para que seja a função que irá calcular os valores seja executada em todos os array selecionados dos elementos. O `if` analisa o a qual categoria objeto pertence e em seguida faz somatória.]

3 referências

Sites

Os sites usados para pesquisa foram:

<https://developer.mozilla.org/pt-BR/>

<https://www.w3schools.com>

Vídeos

durante o processo de desenvolvimento foram assistidos alguns para melhor entendimento do da linguagem.

<https://www.youtube.com/watch?v=w30zWauuoGw&t=1468s>

<https://www.youtube.com/watch?v=IOfDoyP1Aq0&t=2554s>

<https://www.youtube.com/watch?v=JxdsTHdgqAU&t=472s>

Trechos copiados

```
const { error } = require("console");
const fs = require("fs");
const database = require("../broken-database.json");
```

<https://www.youtube.com/watch?v=w30zWauuoGw&t=1468s>

```
///<summary>
/// A OrdenaSaida vai mostrar objetos na ordem estibulada
///</summary>
function OrdenaSaida() {
    // recebe o novo database
    const recebe_JSON = fs.readFileSync("saida.json");
    const dados = JSON.parse(recebe_JSON);

    // o sort para ordenar segundo os parametros
    dados.sort(function (a, b) {
        if (a.category < b.category) {
            return -1;
        } else {
            if (a.category == b.category && a.id < b.id) {
                return -1;
            }
        }
    });
    return console.log(dados);
}
```

<https://www.youtube.com/watch?v=JxdsTHdgqAU&t=472s>

Outras fontes

```
///<summary>
/// a function CorrigeName utiliza o replace para alterar os caracteres que apresentavam erro.
///</summary>
function CorrigeName(name) {
    // define quais caracteres serao mudados e para quais.
    const correcao = { æ: "a", "ç": "c", ß: "b", ø: "o" };
    // /gi: g = global i=ignore
    return name.replace(/æ|ç|ß|ø/gi, function (retorno) {
        return correcao[retorno];
    });
}
```

Graças a um veterano da minha faculdade, que me disponibilizou um código com uma função parecida de corrigir caracteres.