

## TESTE PRÁTICO JS – RM – FELIPE GARBIM

O projeto foi dividido em uma pasta de objetos, uma pasta de serviços, uma pasta `validationSchema`, além do arquivo `resolução.js`.

Foi criada uma service **dataFix**, aonde foi exportada a função `normalize database`, responsável por organizar uma iteração entre os objetos do `broken database`, e através da função **normalizeProduct** realizar a correção de cada objeto iterado com o auxílio das funções `fixName`, `fixQuantity` e `fixPrice`.

É importante observar que dentro das funções `fixName`, `fixQuantity` e `fixPrice` foi realizada a validação dos dados através do **validationSchema**, retornando o dado corrigido, ou em caso de erro encontrado pela validação, um novo objeto com a descrição dos erros encontrados durante a iteração do `database`.

Foi criada uma service para lidar com a gravação dos dados chamada **directory service**, aonde foi utilizada uma função assíncrona afim de lidar com essa atividade custosa da gravação de dados..

**sortService** - Para a ordenação, busquei uma função que realiza a ordenação dos dados em função de duas variáveis.

A Função foi encontrada através do seguinte link:  
["https://stackoverflow.com/questions/13211709/javascript-sort-array-by-multiple-number-fields"](https://stackoverflow.com/questions/13211709/javascript-sort-array-by-multiple-number-fields);

Em **validationSchema** foi utilizada a biblioteca `JOI` para fazer a validação dos dados corrigidos, afim de verificar se houve algum erro durante a normalização dos dados. Link da biblioteca:  
<https://www.npmjs.com/package/joi>

Em **name** ocorreu a validação quando retornado uma string, e verificado se não é nulo.

Em **price** ocorreu a validação, quando retornado um number, verificado se possui 2 casas decimais, se é positivo, e se não é nulo.

E em **quantity** ocorreu a validação quando retornado um number, verificado se é inteiro, e se não é nulo.