

Desafio #2.1

Validação de Arquivo JSON

Instruções

- A aplicação deve ser desenvolvida individualmente.
- Data limite de entrega: 27/07/2023 (quinta-feira).
- A aplicação deve ser versionada e disponibilizada no github.

Descrição

Desenvolver a aplicação **validacao.js** em Javascript (Node.js) usando classes (paradigma OO). A aplicação deverá realizar a validação de um arquivo JSON com o seguinte formato:

```
[ { "nome": "Pedro de Almeida", "cpf": "78444434051",  
  "dt_nascimento": "15021996", "renda_mensal": "3584,12",  
  "estado_civil": "S" },  
  { "nome": "Ana Beatriz", "cpf": "70789422050",  
    "dt_nascimento": "28102000", "renda_mensal": "2810,25",  
    "estado_civil": "S" },  
  { "nome": "Andre Rocha", "cpf": "41719321060",  
    "dt_nascimento": "07051984", "renda_mensal": "4110,39",  
    "estado_civil": "S" },  
  { "nome": "Fernanda Pinheiro", "cpf": "30228380006",  
    "dt_nascimento": "04121972", "renda_mensal": "5850,27",  
    "estado_civil": "C" },  
  { "nome": "Joice Medeiros", "cpf": "31777563097",  
    "dt_nascimento": "21071991", "renda_mensal": "10254,48",  
    "estado_civil": "C" },  
  { "nome": "Carlos Roberto", "cpf": "18493838047",  
    "dt_nascimento": "14091966", "renda_mensal": "8387,92",  
    "estado_civil": "C" } ]
```

Os dados dos clientes devem ser validados usando as seguintes regras:

Campo	Regras	Tipo
Nome	De 5 a 60 caracteres	String
CPF	CPF válido	Number
Data de nascimento	Formato DDMMAAAA O cliente deve ter pelo menos 18 anos na data atual	Date

Renda mensal	Valor Duas casas decimais e vírgula decimal Não obrigatório	Number
Estado civil	C, S, V ou D (maiúsculo ou minúsculo) Não obrigatório	String

Caso haja um ou mais erros em um ou mais clientes, deve ser gerado um arquivo JSON chamado **erros-DDMMAAAA-HHMMSS.json** com o seguinte formato:

```
[ { "dados": { todos os dados do cliente },  
  "erros": [ { "campo": "nome do campo",  
               "mensagem": "mensagem de erro"},  
             { "campo": "nome do campo",  
               "mensagem": "mensagem de erro"},  
             ...  
            ]  
},  
  { "dados": { todos os dados do cliente },  
    "erros": [ { "campo": "nome do campo",  
                  "mensagem": "mensagem de erro"},  
              { "campo": "nome do campo",  
                "mensagem": "mensagem de erro"},  
              ...  
            ]  
},  
  ...  
]
```

Regras

- O arquivo de entrada deverá ser passado como um argumento na execução do programa:
`node validacao.js <path do arquivo JSON de entrada>`
- Caso o arquivo de entrada não exista ou ocorra um erro de leitura, a aplicação deverá apresentar uma mensagem de erro.
- O arquivo de saída deverá ser gerado no mesmo diretório do arquivo de entrada.
- O *timestamp* do nome do arquivo de saída deve ser a data/hora corrente.
- Caso não haja erros de validação, o arquivo de saída deverá ser gerado com o array vazio, ou seja, esse arquivo deve sempre ser gerado.
- Caso ocorra um erro de geração do arquivo de saída, a aplicação deverá apresentar uma mensagem de erro.
- Se o arquivo de entrada não contiver um Array, a aplicação deverá apresentar a mensagem de erro correspondente.
- Caso algum campo obrigatório não seja fornecido, deverá ser gerada a mensagem "campo obrigatório ausente".
- A aplicação deverá ser projetada de forma a separar, no mínimo, as seguintes responsabilidades:

- Leitura do arquivo de entrada
- Validação dos dados
- Geração do arquivo de saída

Dicas

- Para trabalhar com arquivos no Node.js use a API FileSystem:
https://www.w3schools.com/nodejs/nodejs_filesystem.asp
<https://nodejs.dev/pt/learn/writing-files-with-nodejs/>
<https://nodejs.dev/pt/learn/reading-files-with-nodejs/>
- Para trabalhar com data/hora use a API Luxon:
<https://moment.github.io/luxon/#/>
- Para acessar os argumentos de um programa Node.js use o objeto **process**:
https://nodejs.org/docs/latest/api/process.html#process_process_argv

Critérios de Avaliação

- Aplicação do paradigma OO
- Comportamento do programa de acordo com as regras definidas
- Mensagens de erro apropriadas, de acordo com as regras definidas
- Qualidade do código:
 - Indentação
 - Nome de variáveis e métodos
 - Quantidade de parâmetros
 - Separação de responsabilidades