1. Resumo

Esta prova tem como objetivo avaliar o nível de conhecimento do candidato na resolução de problemas utilizando a linguagem C#.

2. Teste

2.1. Desafio 1

Criar um serviço REST (Web API) contenha 2 métodos expostos.

- **AddItemFila**

  O objetivo do método AddItemFila é adicionar um objeto json com o formato abaixo em uma fila de processamento (utilizar o objeto que desejar para o armazenamento).

  **Criar as validações de entrada que achar necessário

  Formato objeto Json de entrada:

```
[
 {
  "moeda": "USD",
  "data_inicio": "2010-01-01",
  "data_fim": "2010-12-01"
 },
 {
  "moeda": "EUR",
  "data_inicio": "2020-01-01",
  "data_fim": "2010-12-01"
 },
 {
  "moeda": "JPY",
  "data_inicio": "2000-03-11",
  "data_fim": "2000-03-30"
 }
]
```

- **GetItemFila**

  O objetivo do método GetItemFila é retornar o último objeto json adicionado na fila pelo método AddItemFila.

  - Caso exista o objeto a ser retornado, retorná-lo e retirá-lo da fila.
  - Caso não exista, retornar algo sinalize que não existe objeto a ser retornado.

2.2. Desafio 2

- **Pré-requisitos**

  Arquivo de entrada: **DadosMoeda.csv** (Será enviado junto com este documento)

  Arquivo de entrada: **DadosCotacao.csv** (Será enviado junto com este documento)

- **Descrição**

Criar uma rotina que rode a cada 2 minutos (Console ou Windows Service) que realize os seguintes passos:

1. Acesse o método **GetItemFila** da api desenvolvida no Item anterior. Caso o método retorne um objeto, obter todas as moedas e períodos correspondentes.
   1.1. Para cada moeda/período retornada da api, acessar o arquivo **DadosMoeda.csv** (mesmo diretório de execução) e trazer todas as moedas/datas que estejam dentro do período (Inclusive) da moeda que está sendo tratada.
   1.2. Com a lista de moedas/datas, buscar todos os valores de cotação (vlr_cotacao) no arquivo **DadosCotacao.csv** utilizando o de-para descrito no **item 4** (Tabela de de-para) para obter as cotações.
   1.3. Gerar um arquivo de resultado (**apenas com as moedas/datas consultadas**) com o nome **Resultado_aaaammdd_HHmmss.csv** no mesmo formato do arquivo **DadosMoeda.csv**. Porém com uma coluna a mais (VL_COTACAO) contendo o valor de cotação correspondente (obtido do arquivo **DadosCotacao.csv**) para cada moeda/data consultada.
   1.4. Encerrar o processamento e aguardar o próximo ciclo de verificação (2 minutos).

2. Caso o método **GetItemFila** da api desenvolvida no item anterior não retorne valor. Encerrar o processamento e aguardar o próximo ciclo de verificação (2 minutos).

3. Observações

- Caso julgue necessário, pode ser gerado um log de processamento da rotina.
- Enquanto o processamento ocorre, o próximo ciclo de processamento (2 minutos) deverá ficar aguardando. **Não pode haver processamento paralelo.
- Utilizar tipos de objetos e lógica de processamento que julguem ser mais performáticas.
- Ao final do processamento (do ciclo) o tempo total de processamento do ciclo deverá ser registrado.
- A prova deverá ser realizada 100% na linguagem C#, sem restrição de versão do .net framework.
- Enviar o link do repositório do teste no github.
- Formato de data do arquivo DadosMoeda.csv: (yyyy-mm-dd)
- Formato de data do arquivo DadosCotacao.csv: (dd/mm/yyyy)

4. Tabela de de-para

DadosCotacao.csv - ID_MOEDA	DadosCotacao.csv - cod_cotacao
AFN	66
ALL	49
ANG	33
ARS	3
AWG	6
BOB	56
BYN	64

CAD	25
CDF	58
CLP	16
COP	37
CRC	52
CUP	8
CVE	51
CZK	29
DJF	36
DZD	54
EGP	12
EUR	20
FJD	38
GBP	22
GEL	48
GIP	18
HTG	63
ILS	40
IRR	17
ISK	11
JPY	9
KES	21
KMF	19
LBP	42
LSL	4
MGA	35
MGB	26
MMK	69
MRO	53
MRU	15
MUR	7
MXN	41
MZN	43
NIO	23
NOK	62
OMR	34
PEN	45
PGK	2
PHP	24
RON	5
SAR	44
SBD	32
SGD	70
SLL	10
SOS	61
SSP	47

SZL	55
THB	39
TRY	13
TTD	67
UGX	59
USD	1
UYU	46
VES	68
VUV	57
WST	28
XAF	30
XAU	60
XDR	27
XOF	14
XPF	50
ZAR	65
ZWL	31