Avaliação desenvolvedor pleno python backend

Shell Script (Linux bash):

Utilizando o arquivo “input” como modelo de dados de entrada, desenvolva shell scripts para o interpretador de comandos bash (exemplos de comandos: echo, awk, cat, etc são permitidos):

1 - Identificar o usuário que tem o maior “size” (Bônus: ser capaz de informar o usuário de menor “size” dado um parâmetro). Abaixo um exemplo:

```
$ ./max-min-size.sh input
juvati_be@uol.com.br inbox 000232478 size 012345671

$ ./max-min-size.sh input -min # Bônus
wjogilabe@uol.com.br inbox 012344022 size 000000008
```

2 - Ordenar o “user” em ordem alfabética (Bônus: capaz de ordenar de forma decrescente dado um parâmetro). Abaixo um exemplo:

```
$ ./order-by-username.sh input| head -n 3
aadmoq@uol.com.br inbox 010030343 size 012115112
aasurari@uol.com.br inbox 000014415 size 002331211
abefadibi@uol.com.br inbox 000141552 size 012042478

$ ./order-by-username.sh input -desc | head -n 3 # Bônus
zzvenaju@uol.com.br inbox 001004028 size 000105267
zzbarica@uol.com.br inbox 001120321 size 000023348
zyzjuf@uol.com.br inbox 002004113 size 000312436
```

3 - Identificar os usuários que estão na seguinte faixa de quantidade de mensagens na INBOX:

```
$ ./between-msgs.sh input 50 200
hgioep@uol.com.br inbox 000000050 size 001003108
vinecitoma@uol.com.br inbox 000000165 size 001214178
rovejusita@uol.com.br inbox 000000153 size 002122412
qivigeva@uol.com.br inbox 000000124 size 001134104
```

Python API

Desenvolver uma API Rest utilizando linguagem Python e um framework de sua escolha. A aplicação terá alguns recursos que irão executar os scripts bash desenvolvidos e converter a saída para JSON conforme exemplo abaixo:

```
# DE:

hgioep@uol.com.br inbox 000000050 size 001003108

// Para:

{"username":"hgioep@uol.com.br","folder":"inbox","numberMessages":50,"size":003108}
```

A solução deverá ter o código compartilhado em um repositório público (github e etc).

Bônus: Implementação de testes automatizados (unitários e/ou integração) e configuração do ambiente no Docker.

1 – Desenvolver um recurso para upload de arquivo, o mesmo só permitirá arquivos de nome com caracteres A-Z, a-z, 0-9, - (hífen) e _ (underline). O arquivo deverá ser salvo em uma pasta que aplicação criará (exemplo: /tmp/teste-api)

MÉTODO	STATUS RETONO
PUT	- 201 (caso o arquivo não exista) - 204 (caso já exista com mesmo nome e foi substituído) - 400 (nome de arquivo não permitido)

2 - Desenvolver um recurso para listagem de arquivos armazenados (Bônus: implementar um mecânismo de paginação):

MÉTODO	STATUS RETONO
GET	- 200 (devolve a lista)

3 – Desenvolver um recurso para obter usuário com tamanho maior size (Bônus: desenvolver também um recurso para menor size), deverá receber o nome de um arquivo que foi armazenado para ser processado utilizando script desenvolvido anteriormente:

MÉTODO	STATUS RETONO
GET	- 200 (devolve o registro) - 404 (arquivo informado não armazenado)

4 - Desenvolver um recurso para obter a lista de usuários ordenados pelo nome de usuário (Bônus: 1 - permitir também ordenar de forma decrescente; 2 - implementar um mecânimos de paginação, 3 – parâmetro para filtro pelo username), deverá receber o nome de um arquivo que foi armazenado para ser processado utilizando script desenvolvido anteriormente:

MÉTODO	STATUS RETONO
GET	- 200 (devolve a lista) - 404 (arquivo informado não existe)

5 - Desenvolver um recurso para obter a lista de usuários entre uma faixa de quantidade de mensagens na INBOX (Bônus: 1 -implementar um mecânimos de paginação; 2 - parâmetro para filtro pelo username), deverá receber o nome de um arquivo que foi armazenado para ser processado utilizando script desenvolvido anteriormente:

MÉTODO	STATUS RETONO
GET	- 200 (devolve a lista) - 404 (arquivo informado não existe)