NodeJS Web Server & File Processing.

El objetivo de este ejercicio es armar una API que pueda devolver métricas obtenidas a partir del procesamiento de un archivo de datos crudo. Los archivos a procesar, serán siempre TSV con las siguientes columnas: id de usuario, lista de segmentos y país. El id de usuario es una string alfanumérica. Cada segmento es un número entero natural, y en la lista se encuentran separados por comas. Por último el país es el código ISO 3166 de dos dígitos.

Por ejemplo, a continuación se incluye un sample de un archivo:

User_id segments country
AxbSDnf 129,345,120,355 AR
nvjsdhfX 2,120,129,10 BR
Sjdksjd2 129,2 AR

La API deberá poder responder las métricas asociadas a un archivo consultado por parámetro, devolviendo la cantidad de usuarios de cada país para cada uno de los segmentos. Por ejempo, para el caso anterior, deberá responderse que el segmento 129 tiene 2 usuarios AR y 1 usuario BR, el 120 tiene 1 AR y 1 BR, el 345 solo 1 AR y así sucesivamente (respetando el formato propuesto a continuación en el ejercicio).

La API también deberá requerir un login con usuario y password, que devolverá un token alfanumérico que será necesario utilizar en los demás endpoints para poder obtener la respuesta. En caso de no enviar el token, los endpoints deberán fallar por Acceso restringido.

1) Descargar los archivos de datos, desde el siguiente repositorio SFTP.
sftp://input.retargetly.com
Path: /files/
User: exercises
Password: gQ6gT72I8bmY
ó
SSH Key: (enviada adjunta).

Y guardarlos en una carpeta interna del web server a realizar.

2) Crear un modelo de datos para guardar la información del siguiente csv:

name,segment1,segment2,segment3,segment4,platformId,clientId
Ricardo,true,false,false,true,1,2
Romina,true,true,true,true,2,3
Joaquin,false,false,false,true,4,4
Lucia,false,false,true,true,10,1
Alejandro,true,false,false,true,1,3
Alicia,true,true,true,true,5,3
Joaquin,false,false,true,false,5,4

Lucia,true,true,true,true,10,1

2) Crear un web server. Este tiene los siguientes endpoints:

A. POST /login -> la api debe permitir a un usuario hacer login. Devolver un token que se utilizar como header "Authorization: [TOKEN]" en cada pedido a los demás endpoints. Deberá haber una política de vencimiento de tokens cada X tiempo.

params:

- user -> usuario
- password -> contraseña

Response:

- Token: (String) Identificador numérico que habiita utilizar los demás endpoints de la API.
- Expires: (Date). Fecha de expiración del token en formato YYYY-MM-DDTHH:MM:SS. Ejemplo: 2019-08-11T00:00:00.

Ejemplo

```
{
    “response”: {
        “token”: “ASdsd2dsdfas3231ds23”,
        “expires”: “2019-08-11T00:00:00”
    }
}
```

B. GET /data -> devuelve en formato JSON los registros del csv del punto 1), fallar si no se recibe el token correcto en el header

params:

- sort -> (string) posibles valores "asc" o "desc". Devuelve la info ordenada de forma ascendente o descendente por el campo especificado en el parámetro sortField
- sortField -> (string) especifica que campo utilizar para ordenar la respuesta de la API
- fields -> (array). Array de los campos a devolver en cada registro. Si no se especifica, se deben devolver todos los campos del csv. Ejemplo: [“name”, “segment1”, “platformId”].
- limit -> (int default 10). Limita la cantidad de registros a devolver.

Response: array con los siguientes atributos

- name, segment1, segment2, segment3, segment4, platformId, clientId

Ejemplo:

```
{
   response: [{
      "name": "Ricardo"
      "platformId": 1
   },{
```

```
    "name": "Romina"
    "platformId": 2
  }]
}
```


C. GET /files/list - Devuelve la lista de archivos disponibles para consultar. Devuelve nombre y peso de los archivos como entero en bytes por defecto pero si se pasa un parámetro humanreadable en true, entonces el peso se devuelve como string redondeando a la unidad más próxima con 1 decimal como máximo. (Ej.: 100 B, 4.5 KB, 25.3 MB, 705 KB, 1.5 GB.

Params:

- humanreadable: true/false (default false)

Response: array con los siguientes atributos

- name: nombre del archivo
- size: peso del archivo

Ejemplo:


```
{
  response: [{
    "name": "archivo 1.tsv"
    "size": 123123
  },{
    "name": "archivo 2.tsv"
    "size": 123123123
  }]
}
```


D. GET /files/metrics: este endpoint deberá devolver las métricas asociadas a cada archivo disponible y el estado de procesamiento del archivo.

params:

- filename -> (string). Filename del archivo del cuál hay que obtener métricas.

Response:

- Status -> (string). started / processing / ready / failed
  - started: significa que se acaba de lanzar el procesamiento para la obtención de las métricas del archivo consultado.
  - processing: el archivo consultado se encuentra siendo procesado en este momento.
  - ready: el archivo ya ha sido procesado y se encuentran disponibles sus métricas.
  - failed: el archivo tiene un formato inválido, no existe, error de procesamiento.
- Started -> (date). Fecha de inicio del proceso del archivo. Formato YYYY-MM-DDTHH:MM:SS. Ejemplo: 2019-08-10T12:23:10.
- Finished -> (date). Fecha de fin del proceso del archivo. Formato YYYY-MM-DDTHH:MM:SS. Ejemplo: 2019-08-10T12:25:10.
- Metrics -> (array). métricas disponibles del archivo. Solo se devuelve si el status = "ready".

- Message -> (string). Solo se devuelve en el caso de status = “failed”, incluir descripción del error (Wrong format, Missing file, etc…).

Ejemplo 1:

```
{
        response: {
                “Status”: “started”,
                “Started”: “2019-08-10T12:23:10”
        }
}
```

Ejemplo 2:

```
{
        response: {
                “Status”: “processing”,
                “Started”: “2019-08-10T12:23:10”
        }
}
```

Ejemplo 3:

```
{
        response: {
                “Status”: “failed”,
                “Message”: “Wrong file format”
        }
}
```

Ejemplo 4:

```
{
        response: {
                “status”: “ready”,
                “started”: “2019-08-10T12:23:10”,
                “finished”: “2019-08-10T12:25:10”,
                “Metrics”: [{
                        “segmentId”: 129,
                        “Uniques”: [{
                                “country”: “AR”,
                                “count”: 2
                        },{
                                “country”: “BR”,
                                “count”: 1
                        }]
                },{
                        “segmentId”: 120,
                        “Uniques”: [{
                                “country”: “AR”,
```

```
                        “count”: 1
                    },{
                        “country”: “BR”,
                        “count”: 1
                    }]
                },{
                    “segmentId”: 345,
                    “Uniques”: [{
                        “country”: “AR”,
                        “count”: 1
                    }]
                },{
                    “segmentId”: 355,
                    “Uniques”: [{
                        “country”: “AR”,
                        “count”: 1
                    }]
                },{
                    “segmentId”: 2,
                    “Uniques”: [{
                        “country”: “AR”,
                        “count”: 1
                    },{
                        “country”: “BR”,
                        “count”: 1
                    }]
                },{
                    “segmentId”: 10,
                    “Uniques”: [{
                        “country”: “BR”,
                        “count”: 1
                    }]
                }]
            }
        }
```


NOTAS:

- Todos los endpoints de la API deberán devolver en el momento (por más que los archivos sean muy pesados para procesar rápido y que esto tarde).
- Todos los endpoints de la API siempre devolverán JSON.