Works Single View (Con Django 3.1.7 y PostgreSQL 13.2)

INSTRUCCIONES: En el archivo requirements.txt, se encuentran las librerias necesarias, para ejecutar el proyecto. La version de Python utilizada es Python 3.9. En el directorio raiz del proyecto /works, donde se encuentra el archivo manage.py, invocar el comando: python3 manage.py runserver(Nota: De ser necesario, vuelva a migrar la base de datos, utilizando los siguientes comandos,
python3 manage.py migrate, y a continuacion, python3 manage.py migrations).

Escriba en su navegador web: 127.0.0.1:8000 o localhost:8000. El procedimiento anterior ejecutara la aplicacion, debera hacer click en seleccionar archivo, y elegir un archivo .csv (en este caso se suministro works_metadata.csv) para ser procesado. Haga click en upload y Works Single View se encargara del resto, mostrando a continuacion el listado de, titulos, autores y ISWC con la informacion procesada, de acuerdo a lo requerido. Puede acceder al panel de administrador de Django escribiendo en su navegador web 127.0.0.1:8000/admin y autenticandose con user: ‘admin’, password: ‘works1234’. En el mismo se puede consultar la base de datos y los registros.

API de Consulta para Works Single Views.

Para utilizar la API de consulta, en el navegador web nos dirigimos al endpoint,
127.0.0.1:8000/musicalworks en el mismo podremos observar un archivo .json con la informacion que disponemos. Arriba a la derecha hacemos click en el boton filters, se nos desplegara un formulario, en el cual vamos a poder realizar la busqueda por ISWC de los metadatos de los que disponemos. Para resumir, pondremos un codigo ISWC y nos devolvera un archivo .json con el titulo, el nombre de los autores el codigo ISWC y el codigo http 200 ok. Mostrando que la solicitud se ha llevado a cabo satisfactoriamente.

Respuestas.

Parte1

1_Se toma el archivo .csv suministrado, y se lo ingresa a Django con request.Files. Chequea, que el archivo tenga una extension .csv. Si no, lo informara a travez de un mensaje de error. Luego, itera sobre la informacion suministrada, linea por linea y separando por titulo, autor e ISWC detectando ','. A continuacion se descartan, si existiesen, los, registros totalmente vacios(los que no tienen ningun dato de titulo, autor e ISWC) y la fila del encabezado (title, contributors ISWC).Se chequea si existe el ISWC, si existe, consultamos la base de datos para ver si ese ISWC ya tiene un registro. Si ya existe ese ISWC, pasamos a chequear el campo titulo, si el campo titulo existe, o sea, no esta vacio, actualizamos nuestra base de datos con la nueva informacion. Despues procedemos a analizar la informacion de los autores, para ello tomamos la informacion de nuestra base de datos y la comparamos con la que queremos introducir. Si es la misma, no hacemos nada. Pero si es diferente, primero unimos la informacion de nuestra base de datos con la nueva a introducir, segundo convertimos la informacion, que es un string separado por '|' en una lista. De esa lista sacamos los elementos repetidos, y luego volvemos a convertir la lista en un string separado por '|' como teniamos, pero ahora con nuestros autores consolidados correctamente. Si al chequear en nuestra base de datos el ISWC, no existiese, creo uno y refactorizo la informacion para consolidar titulo y

contributors en el caso de que existiesen estos ultimos. Por ultimo, si no poseemos en la informacion a actualizar el ISWC, checkeamos si existe el titulo y el autor en la base de datos, si hay registros procedemos a actualizarlo consolidando la informacion.
Habian lineas de codigo que se repetian ya que se requeria hacer la misma tarea repetitiva varias veces. Para mejorar el codigo en legibilidad y comodidad, se creo un modulo , llamado utils, y dentro de el una funcion llamada refactor, encargada de hacer el trabajo. La misma es llamada en varias ocaciones para analizar los campos titulo y autor de la informacion ingresada con la existente en la base de datos. Es la que hace las transformaciones necesarias para consolidar el nombre de los autores de manera correcta.

2_ Se me ocurren dos formas:
La primera, un endpoint en la api donde el cliente envie el .csv y cuando lo manda se procesa (tiene la particularidad que la demanda es cuando el proveedor quiere, y si el fichero es muy grande puede matar la base de datos). Pero se podria solucionar en parte si el proceso es asincrono.
Segunda, forma los clientes dejan el fichero en un servidor, vamos al servidor, nos decargabamos el fichero y lo procesabamos, pero usando un CRON que se ejecute cada una determinada cantidad de timepo que nosotros decidamos (ej. una vez al dia).
Pero resumiendo, un endpoint donde se procesa el archivo asyncronamente para no saturar el servidor o un trabajo con un cron que se ejecuta una vez por dia (o las veces que uno quiera). Vi que existe django-q que es una extension de django para programar trabajos.

Parte2

1_Creo que habria una merma considerable en el rendimiento.

2_Lo primero que se me ocurre es utilizar db_index = True, es una practica frecuente en Django, en los campos que se buscan con frecuencia y que tienen cierto grado de repetición como el ISWC.
Otra opcion seria usar una base de datos no relacional, como MongoDB, ya que e visto en casos donde se reducia a la mitad el tiempo de consulta.
Usar values () y values_list () para seleccionar solo las columnas que desea de una consulta es otra practica comun.
Utilizar el soporte que Django tiene para la búsqueda de texto completo directamente en una base de datos de Postgres.