

Sistemas de Gestion Empresarial

Documentacion Programa Python
Videoclub.py

Jose Luis Lopez Gonzalez

Sistemas de gestión empresarial

Indice:

Tabla de contenido

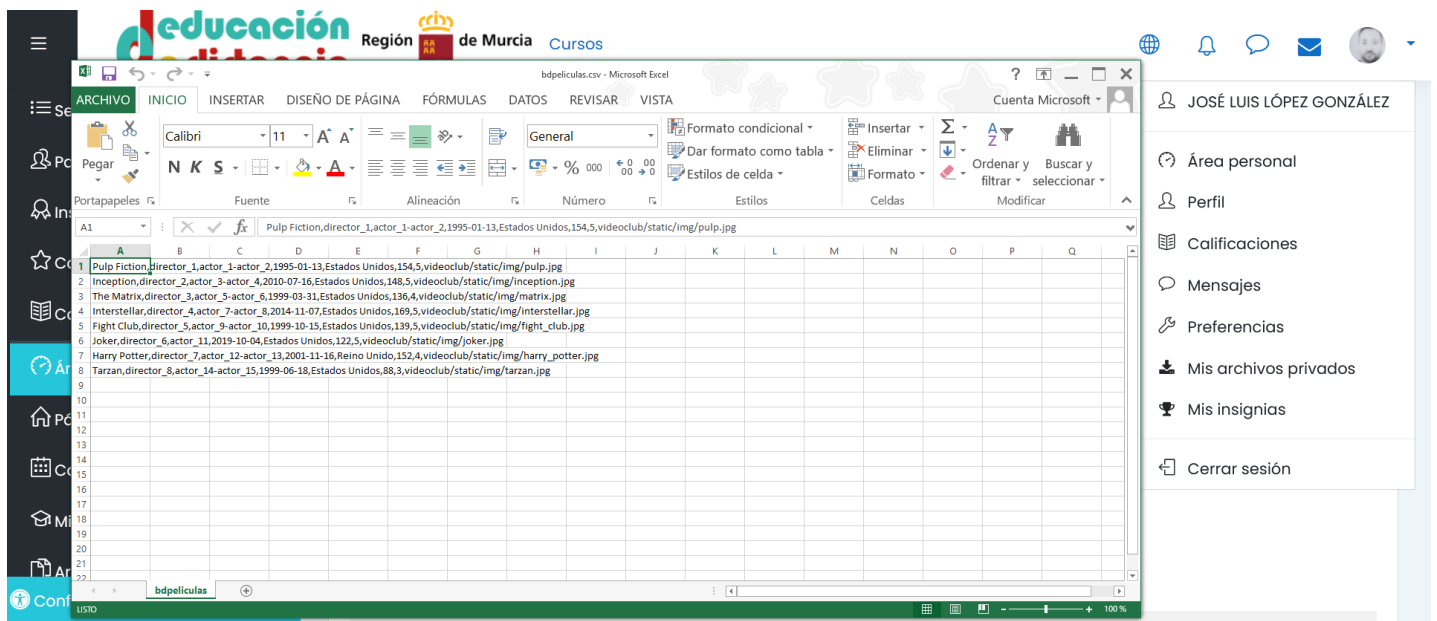
Sistemas de Gestion Empresarial	1
Jose Luis Lopez Gonzalez.....	1
1.Explicacion del programa.	3

Sistemas de gestión empresarial

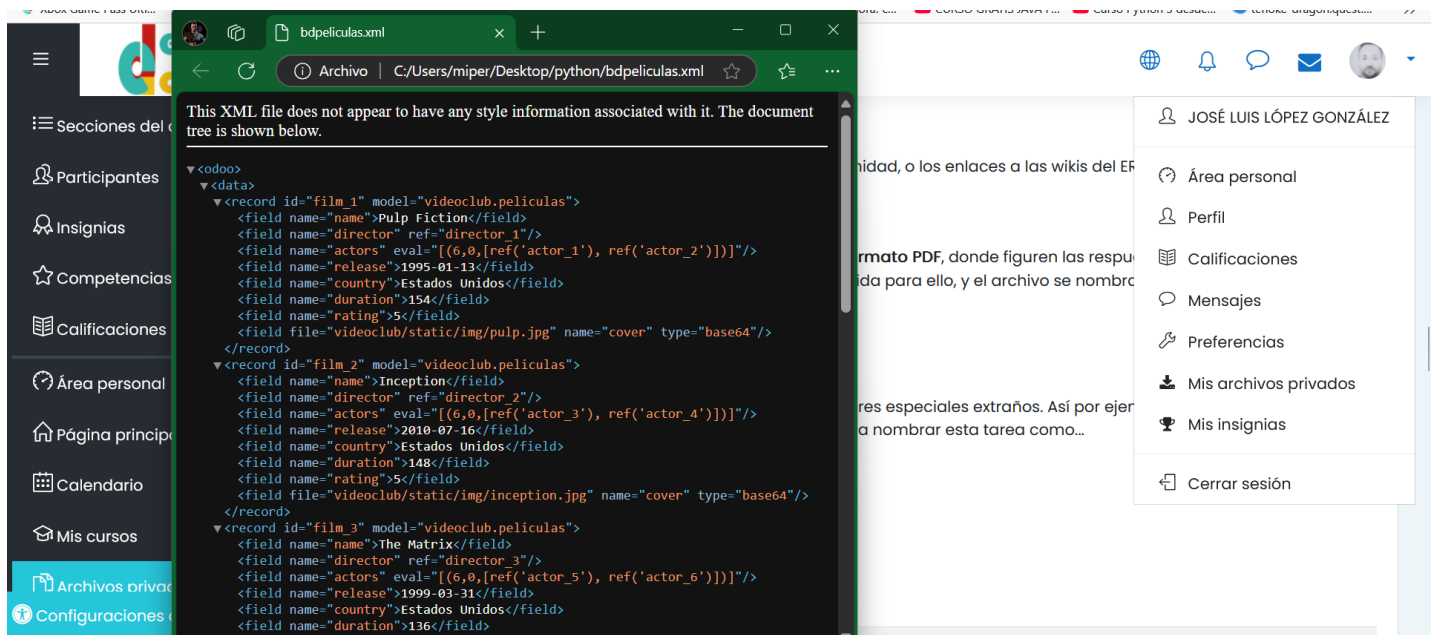
1.Explicacion del programa.

Este programa está diseñado para leer un archivo Csv que contiene datos sobre películas y convertir cada registro en un formato Xml compatible con Odo. El Xml que da de resultado se puede utilizar para importar datos en un sistema de gestión de películas.

Este programa esta compuesto por videoclub.py y peliculas.csv que contiene la base de datos de las películas.



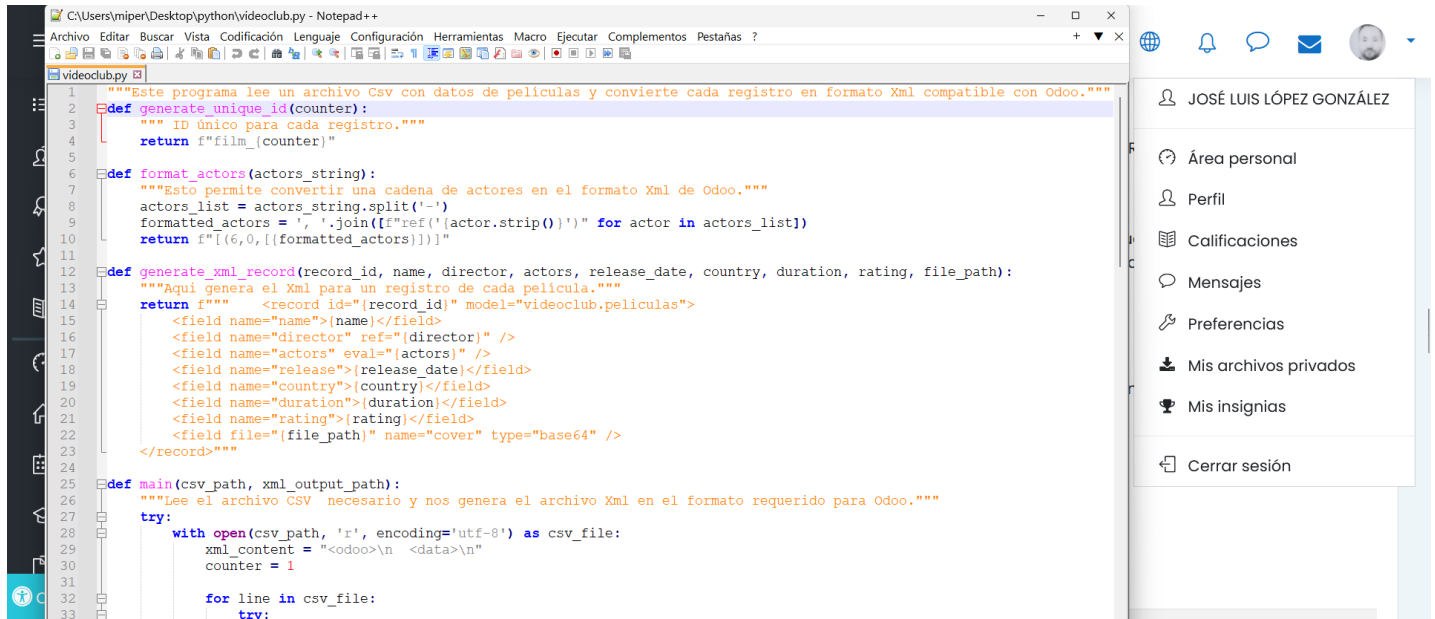
El resultado da un archivo llamado bdpelículas.xml.



Sistemas de gestión empresarial

Las funciones principales son:

`generate_unique_id(counter):`



```
1 """Este programa lee un archivo Csv con datos de películas y convierte cada registro en formato Xml compatible con Odoo."""
2 def generate_unique_id(counter):
3     """ ID único para cada registro."""
4     return f"film_{counter}"
5
6 def format_actors(actors_string):
7     """Esto permite convertir una cadena de actores en el formato Xml de Odoo."""
8     actors_list = actors_string.split('-')
9     formatted_actors = ', '.join([f"ref('{actor.strip()}')" for actor in actors_list])
10    return f"[(6,0,{formatted_actors})]"
11
12 def generate_xml_record(record_id, name, director, actors, release_date, country, duration, rating, file_path):
13    """Aqui genera el Xml para un registro de cada película."""
14    return f"""
15    <record id="{record_id}" model="videoclub.peliculas">
16      <field name="name">{name}</field>
17      <field name="director" ref="{director}" />
18      <field name="actors" eval="{actors}" />
19      <field name="release">{release_date}</field>
20      <field name="country">{country}</field>
21      <field name="duration">{duration}</field>
22      <field name="rating">{rating}</field>
23      <field file="{file_path}" name="cover" type="base64" />
24    </record>"""
25
26 def main(csv_path, xml_output_path):
27    """Lee el archivo CSV necesario y nos genera el archivo Xml en el formato requerido para Odoo."""
28    try:
29        with open(csv_path, 'r', encoding='utf-8') as csv_file:
30            xml_content = "<odoo>\n <data>\n"
31            counter = 1
32
33            for line in csv_file:
34                try:
```

Genera un Id único para cada película y asegura que cada registro tenga un identificador único.

Ejemplo de salida: `film_1`, `film_2`, etc.

`format_actors(actors_string):`

Convierte una cadena de actores en el formato Xml requerido por Odoo, y permite que los actores se referencien de forma correcta en el sistema Odoo.

Ejemplo de salida: `[(6,0,[ref('actor1'), ref('actor2')])]`.

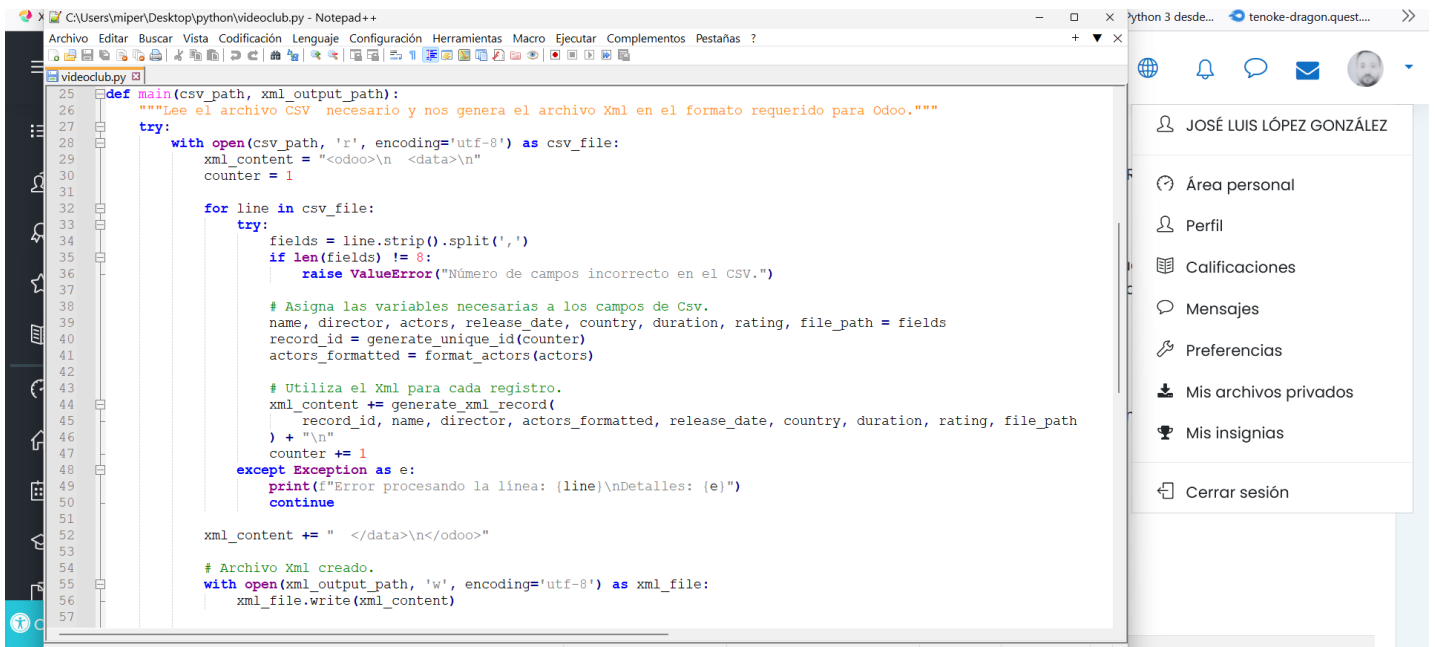
`generate_xml_record(...):`

Crea un bloque Xml para un registro de película utilizando los datos que se le han proporcionado.

Esta función es el núcleo de la generación del Xml, estructurando los datos de manera que Odoo pueda interpretarlos.


Sistemas de gestión empresarial

Funcion Main:



```
25 def main(csv_path, xml_output_path):
26     """Lee el archivo CSV necesario y nos genera el archivo Xml en el formato requerido para Odoo."""
27     try:
28         with open(csv_path, 'r', encoding='utf-8') as csv_file:
29             xml_content = "<odoo>\n <data>\n"
30             counter = 1
31
32             for line in csv_file:
33                 try:
34                     fields = line.strip().split(',')
35                     if len(fields) != 8:
36                         raise ValueError("Número de campos incorrecto en el CSV.")
37
38                     # Asigna las variables necesarias a los campos de Csv.
39                     name, director, actors, release_date, country, duration, rating, file_path = fields
40                     record_id = generate_unique_id(counter)
41                     actors_formatted = format_actors(actors)
42
43                     # Utiliza el Xml para cada registro.
44                     xml_content += generate_xml_record(
45                         record_id, name, director, actors_formatted, release_date, country, duration, rating, file_path
46                     ) + "\n"
47                     counter += 1
48                 except Exception as e:
49                     print(f"Error procesando la línea: {line}\nDetalles: {e}")
50                     continue
51
52             xml_content += " </data>\n</odoo>"
53
54             # Archivo Xml creado.
55             with open(xml_output_path, 'w', encoding='utf-8') as xml_file:
56                 xml_file.write(xml_content)
57
```

Esta función se encarga de todo el proceso: primero, abre y lee el archivo Csv que contiene la información de las películas. Luego, revisa cada línea del archivo para asegurarse de que los datos estén correctos y los separa en partes útiles. A partir de ahí, genera un archivo Xml de salida que organiza toda esa información de manera adecuada.

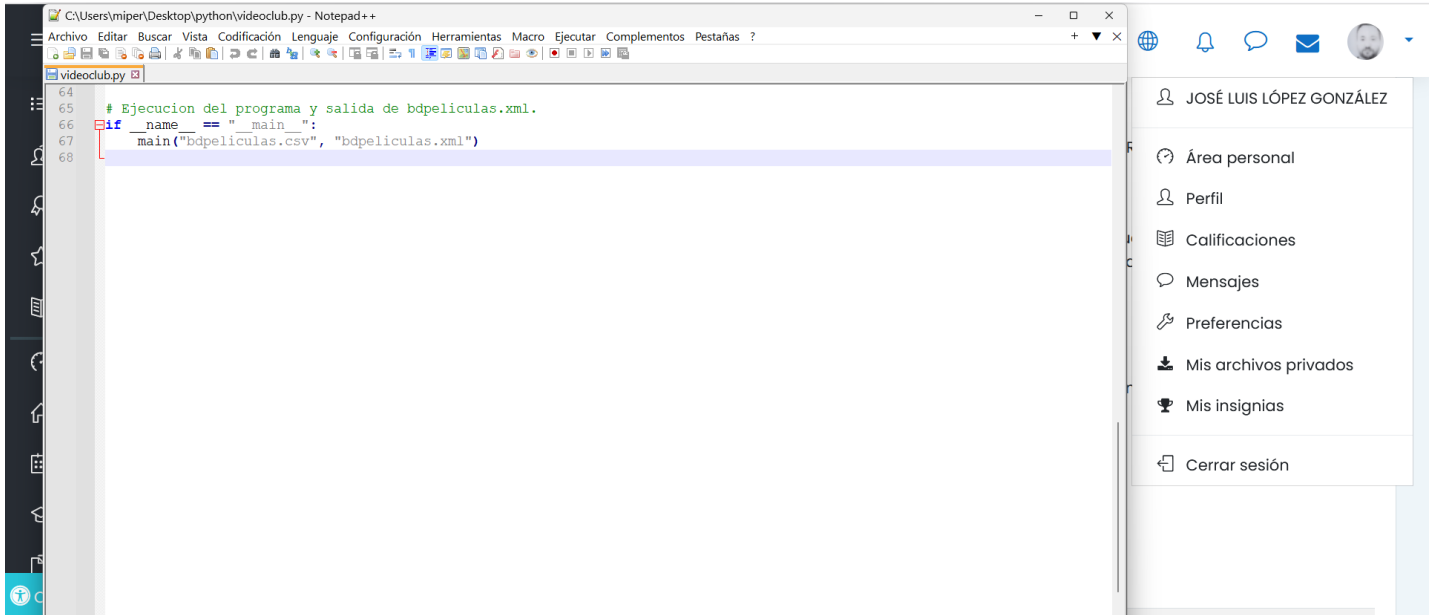


```
27     try:
28         with open(csv_path, 'r', encoding='utf-8') as csv_file:
29             xml_content = "<odoo>\n <data>\n"
30             counter = 1
31
32             for line in csv_file:
33                 try:
34                     fields = line.strip().split(',')
35                     if len(fields) != 8:
36                         raise ValueError("Número de campos incorrecto en el CSV.")
37
38                     # Asigna las variables necesarias a los campos de Csv.
39                     name, director, actors, release_date, country, duration, rating, file_path = fields
40                     record_id = generate_unique_id(counter)
41                     actors_formatted = format_actors(actors)
42
43                     # Utiliza el Xml para cada registro.
44                     xml_content += generate_xml_record(
45                         record_id, name, director, actors_formatted, release_date, country, duration, rating, file_path
46                     ) + "\n"
47                     counter += 1
48                 except Exception as e:
49                     print(f"Error procesando la línea: {line}\nDetalles: {e}")
50                     continue
51
52             xml_content += " </data>\n</odoo>"
53
```

Se utilizan bloques **try-except** para gestionar errores comunes, como la ausencia del archivo CSV o dificultades al procesar las líneas del mismo. Esto permite que el programa maneje situaciones inesperadas de manera más fluida y proporcione mensajes de error claros al usuario.

Sistemas de gestión empresarial

Permite ejecutar el programa dando como resultado bdpelículas.xml.



FIN