

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

SISTEMAS DE BASES DE DATOS 2

SECCIÓN A



Grupo 3 Proyecto 2	
201800476	Marvin Alexis Estrada Florian
201902781	Rodrigo Antonio Porón De León
201800524	Eduardo Tomás Ixén Rucuch

Proceso de carga

En el proceso de carga de este Proyecto 2, consistió en la utilización de información obtenida de **IGDB**, la cual en términos simples es una API la cual provee contenido relacionado con videojuegos, esta información ya había sido almacenada en archivos **.csv** desde el Proyecto 1, ya que se solicitaba la misma información se utilizaron los mismos para la carga de datos, cabe mencionar que estos no fueron directamente cargados a la nueva base de datos en MongoDB, sino que pasaron por un proceso de diseño para su posterior conversión a archivos **.json**.



También se solicitó como requerimiento la obtención específica de datos sobre plataformas, esta información al variar un poco con la información almacenada en el Proyecto 1 si fué necesaria la utilización de la API nuevamente para la obtención de estos nuevos datos, pero eso se detalla más adelante.

- Colección de plataformas

Como se mencionó con anterioridad, a recibir la solicitud de nueva información sobre plataformas, se dió a la tarea de la extracción de datos nuevamente sobre la API, el código utilizado se basa en la utilización de un ciclo que extrae datos de la misma, donde se le indica sobre qué endpoint se quiere la información y que parámetros se desea del mismo, proceso iniciado el 16 de Octubre de 2023 a las 9:36 am y concluyendo con la carga el mismo día a las 6:27 pm.

```
main.py x
Proyecto1 > Extraction_01 > main.py > ...
1 import requests
2 import datetime
3 import time
4 import csv
5 from enum import Enum
6
7 class Region(Enum):
8     europe = 1
9     north_america = 2
10    australia = 3
11    new_zealand = 4
12    japan = 5
13    china = 6
14    asia = 7
15    worldwide = 8
16    korea = 9
17    brazil = 10
18
19 params = {
20     "fields": "company,developer,manufacturer",
21     "limit": 500,
22     "offset": 0
23 }
24
25 def obtain_data():
26     total_data = []
27
28     while True:
29         games = obtain_offset(params['offset'])
30         if not games:
31             break
32
33         total_data.extend(games)
34         params['offset'] += params['limit']
35         time.sleep(1)
36         print(len(total_data))
37
38         #if len(total_data) >= 1000:
39             # break
40
41     # total_data = [game for game in total_data if 'storyline' in game or 'summary' in game]
42     #print(total_data)
43     return total_data
44
45 def convert_bool_to_int(data):
46     for item in data:
47         for key, value in item.items():
48             if isinstance(value, bool):
49                 item[key] = int(value)
50     return data
51
52 def convert_unix_timestamp(timestamp):
53     try:
54         return datetime.datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d')
55     except:
56         return None
57
58 def process_games(games):
59     processed_games = []
60     for game in games:
61         # Convert bool to int
62         game = convert_bool_to_int(game)
63         # Convert unix timestamp to datetime
64         game['release_date'] = convert_unix_timestamp(game['release_date'])
65         # Add to processed games
66         processed_games.append(game)
67     return processed_games
68
69 # Main function
70 def main():
71     # Obtain data
72     total_data = obtain_data()
73     # Convert bool to int
74     total_data = convert_bool_to_int(total_data)
75     # Convert unix timestamp to datetime
76     total_data = convert_unix_timestamp(total_data)
77     # Process games
78     processed_games = process_games(total_data)
79     # Save to CSV
80     save_to_csv(processed_games)
81
82 # Call main function
83 if __name__ == '__main__':
84     main()
```

Esto produce un .csv con la información solicitada, en el formato deseado. Esto no solo fué realizado para la obtención de la nueva información de plataformas de este Proyecto 2, sino que también para toda la información en el Proyecto 1 extraída se realizó de la misma manera, únicamente cambiando el endpoint sobre el cual se deseaba consultar y los parámetros que se necesitaban.

```
main.py x
Proyecto1 > Extraction_01 > main.py > ...
54     return datetime.datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d')
55 except:
56     return None
57
58 def process_games(games):
59     processed_games = []
60     for game in games:
61         if 'game' in game and 'platform' in game:
62             games = game['game']
63             for game_id in games:
64                 processed_game = {'ModoJuego_Id': 2, 'Videojuego_Id': game_id, 'Plataforma_Id': game['platform']}
65                 processed_games.append(processed_game)
66     return processed_games
67
68 def obtain_offset(offset):
69     params['offset'] = offset
70
71     response = requests.post('https://api.igdb.com/v4/regions', headers={'Client-ID': 'kk13orxbx5oyw1ryu192wc4xqkejrp', 'Autho
72
73 if response.status_code == 200:
74     return response.json()
75 else:
76     print(f"Error: {response.status_code} - {response.text}")
77     return []
78
79 def write(data):
80     with open("regions.csv", mode="w", newline="", encoding="utf-8") as csv_file:
81         all_columns = set()
82
83         for row in data:
84             all_columns.update(row.keys())
85
86         writer = csv.DictWriter(csv_file, fieldnames=list(all_columns))
87
88         writer.writeheader()
89
90         for row in data:
91             writer.writerow(row)
92
93     print("Data saved")
94
95 def data_count():
96     response = requests.post('https://api.igdb.com/v4/regions/count', headers={'Client-ID': 'kk13orxbx5oyw1ryu192wc4xqkejrp',
97
98 if response.status_code == 200:
99     return response.json()
100 else:
101     print(f"Error: {response.status_code} - {response.text}")
102     return []
103
104 def main():
105     print(data_count())
106     data = obtain_data()
107     write(data)
108
109 if __name__ == "__main__":
110     main()
111
```

Estos son algunos ejemplos de la extracción de datos de la API del Proyecto 2 y utilizados para la obtención de la nueva información de plataformas:

- Plataformas_alpha

```
plataformas_alpha.csv X
Proyecto2 > data > plataformas_alpha.csv
1 category,generation,name,abbreviation,platform_family,id,alternative_name,summary,versions
2 6,,Commodore CDTV,,158,Commodore Dynamic Total Vision,,[223]
3 1,4,Sega Pico,,3,339,Kids Computer Pico,,[456]
4 1,6,PlayStation 2,PS2,1,8,PS2,,"[58, 114]"
5 4,,iOS,iOS,,39,,[43]
6 6,,Commodore Plus/4,C+4,,94,,[108]
7 1,,AY-3-8710,,144,,[207]
8 1,1,Odyssey,odyssey,,88,Magnavox Odyssey; Odysee; Odisa; Odissea,,"[101, 167, 168, 169, 170, 171]"
9 6,,Commodore PET,cpet,90,,[103]
10 6,,Sol-20,,237,,[354]
11 4,,PC (Microsoft Windows),PC,,6,mwin,,"[1, 13, 14, 15, 124]"
12 5,,Tapwave Zodiac,zod,44,,[69]
13 1,2,ColecoVision,colecovision,,68,,[35]
14 6,,Texas Instruments TI-99,ti-99,,129,Texas Instruments TI-99/4A,,"[172, 427]"
15 6,,Acorn Electron,,134,,[184]
16 5,4,Gamate,,378,Super Boy,,[497]
17 2,,Hyper Neo Geo 64,,135,,[186]
18 6,,Thomson MO5,,156,,[221]
19 6,,Odyssey 2 / Videopac G7000,,133,Magnavox Odyssey2,,[183]
20 ,,SteamVR,Steam VR,,163,,[233]
21 1,1,PC-50X Family,,142,,,"[200, 205]"
22 1,,AY-3-8607,,148,,[211]
23 1,,AY-3-8605,,146,,[209]
24 1,,AY-3-8606,,147,,[210]
25 6,,Amstrad CPC,ACPC,,25,Colour Personal Computer,,[20]
26 5,9,Playdate,,381,,[501]
```

- Plataformas_versiones

```
plataformas_versiones.csv X
Proyecto2 > data > plataformas_versiones.csv
1 platform_version_release_dates,companies,main_manufacturer,id,summary,name
2 [552],[533],,103,"The Commodore PET is a line of home/personal computers produced starting
3 [359],[527],,282,"A smaller version of the Switch, which only supports handheld play. The 3
4 [204],[144],,139,Original Japanese version (SHVC-001),Super Famicom (SHVC-001)
5 ,,164,,Initial version
6 ,,222,,Initial version
7 "[360, 361, 362]",[368],,283,"A smaller version of the Wii with a top-loading disc drive. T
8 "[36, 37, 39, 133]",,51,"In the early 1990s there was a trend by video game companies to u
9
10 Chairman of Nintendo of America, Howard Lincoln, teamed with Jim Clark, Chairman of Silicon
11
12 On July 18, 1994, Nintendo revealed that the new name for their system, the ""Ultra 64"" (1
13
14 Nintendo announced that it would release the Nintendo 64 system to American consumers on Ap
15
16 When the Nintendo 64 system was finally launched on September 29, 1996, only two games were
17
18 The Nintendo 64 system and the game Super Mario 64 opened to rave reviews from the critics.
19
20 On its first day of sale, the Nintendo 64 system sold 500,000 units in North America. It to
```

Luego se procedió a la utilización de estos csv's, sobre el cual con ese código se realizó el cambio de formato a .json para poder importarlos posteriormente a mongodb

```
platforms.py X
Proyecto2 > platforms.py > ...
1 import csv
2 import json
3
4 # ENUM para tipo
5 tipo_enum = {
6     "1": "console",
7     "2": "arcade",
8     "3": "platform",
9     "4": "operating_system",
10    "5": "portable_console",
11    "6": "computer"
12 }
13
14 # ENUM para familia
15 familia_enum = {
16     "5": "Nintendo",
17     "4": "Linux",
18     "2": "Xbox",
19     "3": "Sega",
20     "1": "PlayStation"
21 }
22
23 # Mapeo de regiones según el enum
24 region_enum = {
25     "1": "europe",
26     "2": "north_america",
27     "3": "australia",
28     "4": "new_zealand",
29     "5": "japan",
30     "6": "china",
31     "7": "asia",
32     "8": "worldwide",
33     "9": "korea",
34     "10": "brazil"
35 }
36
37 data = []
38
39 # Crea un diccionario para almacenar todos los datos del archivo "plataformas_alpha.csv"
40 csv_data = {}
41
42 # Abre el archivo "plataformas_alpha.csv" y lee los datos
43 with open("./data/plataformas_alpha.csv", mode="r", encoding="utf-8") as csv_file:
44     csv_reader = csv.DictReader(csv_file)
45
46     for row in csv_reader:
47         csv_data[row["id"]] = row
48
49 # Crea un diccionario para almacenar todos los datos del archivo "plataformas_versiones.csv"
50 versions_data = {}
51
52 # Abre el archivo "plataformas_versiones.csv" y lee los datos
53 with open("./data/plataformas_versiones.csv", mode="r", encoding="utf-8") as versions_file:
54     versions_reader = csv.DictReader(versions_file)
55
56     for row in versions_reader:
57         versions_data[row["id"]] = row
58
```

```

platform.py X
Proyecto2 > platforms.py > ...
117
118 # Agrega nombre alternativo si está presente
119 alternative_name = row.get("alternative_name")
120 if alternative_name:
121     json_data["nombre_alternativo"] = alternative_name
122
123 # Procesa el primer elemento del array "versions" para buscar fechas de lanzamiento y descripción
124 versions_str = row.get("versions")
125 if versions_str:
126     versions = json.loads(versions_str.replace("'", "\'"))
127     if versions and len(versions) > 0:
128         version_id = versions[0]
129         if str(version_id) in versions_data:
130             platform_version = versions_data[str(version_id)]
131             platform_version_release_dates_str = platform_version.get("platform_version_release_dates")
132             if platform_version_release_dates_str:
133                 release_date_ids = json.loads(platform_version_release_dates_str.replace("'", "\'"))
134                 if release_date_ids:
135                     release_dates = []
136                     for release_date_id in release_date_ids:
137                         if str(release_date_id) in release_date_data:
138                             release_date_info = release_date_data[str(release_date_id)]
139                             region = region_enum.get(release_date_info["region"]).lower(), release_date_info["region"]
140                             release_dates.append({"fecha": release_date_info["fecha"], "region": region})
141                 if release_dates:
142                     json_data["fechas_lanzamiento"] = release_dates
143
144 # Obtiene la descripción desde el primer elemento del array "versions"
145 version_description = platform_version.get("summary")
146 if version_description:
147     json_data["descripcion"] = version_description
148
149 # Busca empresas asociadas
150 version_companies_str = platform_version.get("companies")
151 if version_companies_str:
152     version_companies_ids = json.loads(version_companies_str.replace("'", "\'"))
153     if version_companies_ids:
154         company_names = []
155         for company_id in version_companies_ids:
156             company_id = str(company_id)
157             if company_id in version_companies_data:
158                 company_id_in_companies = version_companies_data[company_id]["company"]
159                 if company_id_in_companies in company_data:
160                     company_names.append({"nombre": company_data[company_id_in_companies]["company_name"]})
161         if company_names:
162             json_data["empresas"] = company_names
163
164 data.append(json_data)
165
166 # Escribe los datos en un archivo JSON
167 with open("plataformas_alpha.json", mode="w", encoding="utf-8") as json_file:
168     json.dump(data, json_file, ensure_ascii=False, indent=4)
169
170 print("Datos convertidos y guardados en plataformas_alpha.json")
171

```

Por último, al .json de plataformas se le añadió los juegos que posee cada una de las mismas con el siguiente código.

```

platforms_game.py X
Proyecto2 > platforms_game.py > ...
1 import json
2 import csv
3
4 # Cargar datos de Videojuego_plataforma.csv y crear un diccionario de juegos por plataforma
5 juegos_por_plataforma = {}
6 with open('./data/Videojuego_plataforma.csv', 'r', encoding='utf-8') as csv_file:
7     csv_reader = csv.DictReader(csv_file)
8     for row in csv_reader:
9         platform_id = int(row['platform_id'])
10         game_id = int(row['game_id'])
11         fecha = row['date']
12         region = row['region']
13         juegos_por_plataforma.setdefault(platform_id, []).append((game_id, fecha, region))
14
15 # Cargar datos de juegos_sin_parrafos.csv y crear un diccionario de nombres de juegos por igdb_id
16 nombres_de_juegos = {}
17 with open('./data/juegos_sin_parrafos.csv', 'r', encoding='utf-8') as csv_file:
18     csv_reader = csv.DictReader(csv_file)
19     for row in csv_reader:
20         igdb_id = int(row['igdb_id'])
21         nombre = row['nombre']
22         nombres_de_juegos[igdb_id] = nombre
23
24 # Cargar el archivo plataformas_alpha.json
25 with open('plataformas_alpha.json', 'r', encoding='utf-8') as json_file:
26     data = json.load(json_file)
27
28 # Actualizar el atributo "juegos" en el archivo JSON
29 for plataforma in data:
30     plataforma_id = plataforma['_id']
31     if plataforma_id in juegos_por_plataforma:
32         juegos_de_plataforma = []
33         for game_id, _, _ in juegos_por_plataforma[plataforma_id]:
34             juegos_de_plataforma.append(game_id)
35     plataforma['juegos'] = juegos_de_plataforma
36
37 # Guardar el nuevo JSON
38 with open('plataformas_actualizado_ids.json', 'w', encoding='utf-8') as json_file:
39     json.dump(data, json_file, ensure_ascii=False, indent=4)
40

```

Luego de todo el proceso mencionado anteriormente, se tiene la producción del json con plataformas, el cual fué importado a MongoDB en el siguiente formato:

```
{} plataformas_actualizado.json X
Proyecto2 > data_02 > {} plataformas_actualizado.json
1  [
2  {
3    "_id": 158,
4    "igdb_platform_id": 158,
5    "nombre": "Commodore CDTV",
6    "tipo": "computer",
7    "nombre_alternativo": "Commodore Dynamic Total Vision",
8    "fechas_lanzamiento": [
9      {
10       "fecha": "Mar 03, 1991",
11       "region": "north_america"
12     },
13     {
14       "fecha": "Nov 14, 1991",
15       "region": "europe"
16     },
17     {
18       "fecha": "Jul 12, 1991",
19       "region": "australia"
20     }
21   ],
22   "descripcion": "The CDTV is essentially a Commodore Amiga 500 home computer with a CD-ROM drive and remote control. With the optional key",
23   "juegos": [
24     {
25       "igdb_id": 38843,
26       "nombre": "Prey: An Alien Encounter",
27       "fecha": "1994-12-30",
28       "region": "worldwide"
29     },
30     {
31       "igdb_id": 11974,
32       "nombre": "Body Blows",
33       "fecha": "1993-02-28",
34       "region": "worldwide"
35     },
36     {
37       "igdb_id": 66345,
38       "nombre": "Snoopy: The Cool Computer Game",
39       "fecha": "1991-12-30",
40       "region": "north_america"
41     },
42     {
43       "igdb_id": 12832,
44       "nombre": "Wrath of the Demon",
45       "fecha": "1991-12-30",
46       "region": "worldwide"
47     },
48     {
49       "igdb_id": 69874,
50       "nombre": "Town With No Name",
51       "fecha": "1992-12-30",
52       "region": "worldwide"
53     },
54     {
55       "igdb_id": 2603,
56       "nombre": "Battle Chess",
57       "fecha": "1992-12-30",
58       "region": "north_america"
59     }
60   ]
61 }
```


- Colección de juegos

Esta información ya se encontraba en los archivos **.csv**, por lo que únicamente se procedió al procesamiento de datos y su respectiva conversión a **.json** con el siguiente código:

```
game.py X
Proyecto2 > game.py > ...
1 import pandas as pd
2 import json
3
4 # Leer el archivo "juegos_sin_parrafos" en un DataFrame de pandas
5 juegos_df = pd.read_csv("./data/juegos_sin_parrafos.csv")
6
7 # Leer los archivos CSV de "motores" y "Videojuego_Motor"
8 motores_df = pd.read_csv("./data/motores.csv")
9 videojuego_motor_df = pd.read_csv("./data/Videojuego_Motor.csv")
10
11 # Leer el archivo CSV de "Videojuego_Tema"
12 videojuego_tema_df = pd.read_csv("./data/Videojuego_Tema.csv")
13
14 # Leer el archivo CSV de "temas"
15 temas_df = pd.read_csv("./data/temas.csv")
16
17 # Leer el archivo "series.csv"
18 series_df = pd.read_csv("./data/series.csv")
19
20 # Leer el archivo "tipos.csv"
21 tipos_df = pd.read_csv("./data/tipos.csv")
22
23 # Leer el archivo "franquicias.csv"
24 franquicias_df = pd.read_csv("./data/franquicias.csv")
25
26 # Leer el archivo "empresas.csv"
27 empresas_df = pd.read_csv("./data/empresas.csv")
28
29 # Leer el archivo "Videojuego_Empresa.csv"
30 videojuego_empresa_df = pd.read_csv("./data/Videojuego_Empresa_02.csv")
31
32 # Leer el archivo "Videojuego_Genero.csv"
33 videojuego_genero_df = pd.read_csv("./data/Videojuego_Genero.csv")
34
35 # Leer el archivo "generos.csv"
36 generos_df = pd.read_csv("./data/generos.csv")
37
38 # Leer el archivo "Videojuego_Perspectiva.csv"
39 videojuego_perspectiva_df = pd.read_csv("./data/Videojuego_Perspectiva.csv")
40
41 # Leer el archivo "perspectivas.csv"
42 perspectivas_df = pd.read_csv("./data/perspectivas.csv")
43
44 # Leer el archivo "Videojuego_CatEdad.csv"
45 videojuego_catedad_df = pd.read_csv("./data/Videojuego_CatEdad.csv")
46
47 # Leer el archivo "cat_edad.csv"
48 cat_edad_df = pd.read_csv("./data/cat_edad.csv")
49
50 # Leer el archivo "titulos.csv"
51 titulos_df = pd.read_csv("./data/titulos.csv")
52
53 # Realizar la unión de datos en función de igdb_id
54 result_df = pd.merge(juegos_df, videojuego_motor_df, left_on="igdb_id", right_on="Videojuego_Id", how="left")
55
56 # Agregar los nombres de los motores en función de Motor_Id
57 result_df = pd.merge(result_df, motores_df, left_on="Motor_Id", right_on="id", how="left")
58
```

```

game.py X
Proyecto2 > game.py > ...
157     documento["motores"] = motores
158
159     # Agregar empresas sin duplicados
160     empresas = [{"nombre": nombre} for nombre in set(group["company_name"].dropna().values)]
161     if empresas:
162         documento["empresas"] = empresas
163
164     # Agregar géneros sin duplicados
165     generos = [{"tipo_genero": tipo} for tipo in set(group["tipo_genero"].dropna().values)]
166     if generos:
167         documento["generos"] = generos
168
169     # Agregar temas sin duplicados
170     temas = [{"tipo": tipo} for tipo in set(group["name_y"].dropna().values)]
171     if temas:
172         documento["temas"] = temas
173
174     # Agregar perspectivas sin duplicados
175     perspectivas = [{"tipo_perspectiva": tipo} for tipo in set(group["tipo_perspectiva"].dropna().values)]
176     if perspectivas:
177         documento["perspectivas"] = perspectivas
178
179     # Agregar categorías de edad sin duplicados
180     catedades = [{"tipo_cat_edad": tipo} for tipo in set(group["tipo_cat_edad"].dropna().values)]
181     if catedades:
182         documento["categorias_edad"] = catedades
183
184     # Agregar títulos con comentarios sin duplicados y sin valores NaN
185     titulos = []
186     titulos_set = set() # Conjunto para rastrear títulos duplicados
187     for _, row in group.iterrows():
188         titulo_nombre = row["title"]
189         comentario = row["comment"]
190         if not pd.isna(titulo_nombre) or not pd.isna(comentario):
191             if titulo_nombre not in titulos_set:
192                 titulo = {}
193                 if not pd.isna(titulo_nombre):
194                     titulo["nombre"] = titulo_nombre
195                 if not pd.isna(comentario):
196                     titulo["comentario"] = comentario
197                 titulos.append(titulo)
198                 titulos_set.add(titulo_nombre)
199     if titulos:
200         documento["titulos"] = titulos
201
202     documentos.append(documento)
203
204     # Guardar el resultado en un archivo JSON final
205     with open("juegos_con_franquicias.json", "w") as json_file:
206         json.dump(documentos, json_file, indent=4)
207
208     print("Archivo JSON 'juegos_con_motores_empresas_generos_temas_serie_perspectivas_categorias_edad_titulos.json' creado con éxito.")

```

Esto agrega casi la totalidad de datos solicitados sobre los videojuegos, lo único que no poseen son plataformas e idiomas, los cuales se utilizó un proceso diferente para producir el **.json** final, proceso iniciado el 13 de Octubre de 2023 a las 10:12 am y concluyendo con la carga el 15 de Octubre a las 5:47 pm..

Con este procedimiento o código se insertan las respectivas plataformas directamente al **.json** de salida del proceso anterior.

```
game_platforms.py X
Proyecto2 > game_platforms.py > ...
1 import json
2 import csv
3
4 # Leer el archivo "plataformas.csv" y crear un diccionario que mapee IDs de plataforma a nombres
5 plataformas_dict = {}
6 with open("./data/plataformas.csv", "r") as plataformas_file:
7     reader = csv.reader(plataformas_file)
8     next(reader) # Saltar la primera línea con encabezados
9     for row in reader:
10         plataforma_id = int(row[1])
11         plataforma_nombre = row[0]
12         plataformas_dict[plataforma_id] = plataforma_nombre
13
14 # Leer el archivo "Videojuego_Plataforma.csv" y crear un diccionario que mapee IDs de juegos a plataformas y datos de lanzamiento
15 plataformas_por_juego = {}
16 with open("./data/Videojuego_Plataforma.csv", "r") as plataformas_juego_file:
17     reader = csv.reader(plataformas_juego_file)
18     next(reader) # Saltar la primera línea con encabezados
19     for row in reader:
20         juego_id = int(row[0])
21         plataforma_id = int(row[1])
22         fecha = row[2]
23         region = row[3]
24
25         # Crear un diccionario para representar los datos de lanzamiento
26         datos_lanzamiento = {"fecha": fecha, "region": region}
27
28         # Agregar la plataforma y datos de lanzamiento al juego correspondiente
29         if juego_id not in plataformas_por_juego:
30             plataformas_por_juego[juego_id] = []
31         if plataforma_id in plataformas_dict:
32             plataformas_por_juego[juego_id].append({"nombre": plataformas_dict[plataforma_id], "datos_lanzamiento": datos_lanzamiento})
33         else:
34             print(f"Plataforma ID {plataforma_id} no encontrado en el diccionario.")
35
36 # Leer el JSON original
37 with open("./data_02/games_with_languages.json", "r") as json_file:
38     juegos_json = json.load(json_file)
39
40 # Función para agregar plataformas a un juego en el formato deseado
41 def agregar_plataformas_a_juego(juego):
42     igdb_id = juego["igdb_id"]
43     plataformas_juego = plataformas_por_juego.get(igdb_id, [])
44     plataformas_formateadas = []
45
46     # Crear un diccionario para agrupar lanzamientos por nombre de plataforma
47     plataformas_dict = {}
48
49     for p in plataformas_juego:
50         nombre_plataforma = p["nombre"]
51         lanzamiento = {"fecha": p["datos_lanzamiento"]["fecha"], "region": p["datos_lanzamiento"]["region"]}
52
53         if nombre_plataforma not in plataformas_dict:
54             plataformas_dict[nombre_plataforma] = []
55
56         plataformas_dict[nombre_plataforma].append(lanzamiento)
57
58     # Formatear las plataformas
59     for nombre, lanzamientos in plataformas_dict.items():
60         plataforma_formateada = {"nombre": nombre, "lanzamientos": lanzamientos}
61         plataformas_formateadas.append(plataforma_formateada)
62
63     # Validación: No agregar "plataformas" si el array está vacío
64     if plataformas_formateadas:
65         juego["plataformas"] = plataformas_formateadas
66     elif "plataformas" in juego:
67         del juego["plataformas"]
68
69 # Agregar plataformas al formato deseado a cada juego
70 cont = 1
71 for juego in juegos_json:
72     print(cont)
73     cont += 1
74     agregar_plataformas_a_juego(juego)
75
76 # Guardar el resultado en un nuevo archivo JSON
77 with open("./data_02/games_with_platforms.json", "w") as json_file:
78     json.dump(juegos_json, json_file, indent=4)
79
80 print('Archivo JSON "games_with_platforms.json" creado con éxito.')
```

```
game_platforms.py X
Proyecto2 > game_platforms.py > ...
38 with open("./data_02/games_with_languages.json", "r") as json_file:
39     juegos_json = json.load(json_file)
40
41 # Función para agregar plataformas a un juego en el formato deseado
42 def agregar_plataformas_a_juego(juego):
43     igdb_id = juego["igdb_id"]
44     plataformas_juego = plataformas_por_juego.get(igdb_id, [])
45     plataformas_formateadas = []
46
47     # Crear un diccionario para agrupar lanzamientos por nombre de plataforma
48     plataformas_dict = {}
49
50     for p in plataformas_juego:
51         nombre_plataforma = p["nombre"]
52         lanzamiento = {"fecha": p["datos_lanzamiento"]["fecha"], "region": p["datos_lanzamiento"]["region"]}
53
54         if nombre_plataforma not in plataformas_dict:
55             plataformas_dict[nombre_plataforma] = []
56
57         plataformas_dict[nombre_plataforma].append(lanzamiento)
58
59     # Formatear las plataformas
60     for nombre, lanzamientos in plataformas_dict.items():
61         plataforma_formateada = {"nombre": nombre, "lanzamientos": lanzamientos}
62         plataformas_formateadas.append(plataforma_formateada)
63
64     # Validación: No agregar "plataformas" si el array está vacío
65     if plataformas_formateadas:
66         juego["plataformas"] = plataformas_formateadas
67     elif "plataformas" in juego:
68         del juego["plataformas"]
69
70 # Agregar plataformas al formato deseado a cada juego
71 cont = 1
72 for juego in juegos_json:
73     print(cont)
74     cont += 1
75     agregar_plataformas_a_juego(juego)
76
77 # Guardar el resultado en un nuevo archivo JSON
78 with open("./data_02/games_with_platforms.json", "w") as json_file:
79     json.dump(juegos_json, json_file, indent=4)
80
81 print('Archivo JSON "games_with_platforms.json" creado con éxito.')
```

Como último paso se procede a agregar los lenguajes a cada videojuego, utilizando en **.json** resultante del código anterior de plataformas.

```
game_languages.py X
Proyecto2 > game_languages.py > ...
1 import json
2
3 # Crear un diccionario con la información de idiomas
4 idiomas_dict = {
5     1: "Arabic",
6     2: "Chinese (Simplified)",
7     3: "Chinese (Traditional)",
8     4: "Czech",
9     5: "Danish",
10    6: "Dutch",
11    7: "English",
12    8: "English (UK)",
13    9: "Spanish (Spain)",
14    10: "Spanish (Mexico)",
15    11: "Finnish",
16    12: "French",
17    13: "Hebrew",
18    14: "Hungarian",
19    15: "Italian",
20    16: "Japanese",
21    17: "Korean",
22    18: "Norwegian",
23    19: "Polish",
24    20: "Portuguese (Portugal)",
25    21: "Portuguese (Brazil)",
26    22: "Russian",
27    23: "Swedish",
28    24: "Turkish",
29    25: "Thai",
30    26: "Vietnamese",
31    27: "German",
32    28: "Ukrainian"
33 }
34
35 # Crear un diccionario con la información de tipos de soporte
36 tipo_soporte_dict = {
37     1: "Audio",
38     2: "Subtitles",
39     3: "Interface"
40 }
41
42 # Crear un diccionario para mapear IDs de juegos a listas de idiomas
43 idiomas_por_juego = {}
44
45 # Leer el archivo "Videojuego_Idioma.csv" y crear diccionarios que mapeen IDs a nombres
46 with open("./data/Videojuego_Idioma.csv", "r") as idiomas_file:
47     next(idiomas_file) # Saltar la primera línea con encabezados
48     for line in idiomas_file:
49         partes = line.strip().split(",")
50         igdb_id = int(partes[1])
51         id_idioma = int(partes[0])
52         tipo_soporte_id = int(partes[2])
53         if igdb_id not in idiomas_por_juego:
54             idiomas_por_juego[igdb_id] = []
55         idiomas_por_juego[igdb_id].append({"nombre": id_idioma, "tipo": tipo_soporte_id})
56
57 # Leer el JSON original
58 with open("./data_02/games.json", "r") as json_file:
59     juegos_json = json.load(json_file)
```

game_languages.py X

Proyecto2 > game_languages.py > ...

```
51         id_idioma = int(partes[0])
52         tipo_soporte_id = int(partes[2])
53         if igdb_id not in idiomas_por_juego:
54             idiomas_por_juego[igdb_id] = []
55         idiomas_por_juego[igdb_id].append({"nombre": id_idioma, "tipo": tipo_soporte_id})
56
57 # Leer el JSON original
58 with open("../data_02/games.json", "r") as json_file:
59     juegos_json = json.load(json_file)
60
61 # Función para agregar idiomas a un juego en el formato deseado
62 def agregar_idiomas_a_juego(juego):
63     igdb_id = juego["igdb_id"]
64     idiomas_juego = idiomas_por_juego.get(igdb_id, [])
65     idiomas_formateados = {}
66
67     for idioma in idiomas_juego:
68         nombre_idioma = idiomas_dict.get(idioma["nombre"], "Desconocido")
69         tipo_soporte = tipo_soporte_dict.get(idioma["tipo"], "Desconocido")
70
71         if nombre_idioma not in idiomas_formateados:
72             idiomas_formateados[nombre_idioma] = {"nombre": nombre_idioma, "implementaciones": []}
73
74         idiomas_formateados[nombre_idioma]["implementaciones"].append({"tipo": tipo_soporte})
75
76 # Filtra y elimina los idiomas con arrays vacíos
77 idiomas_filtrados = [idioma for idioma in idiomas_formateados.values() if idioma["implementaciones"]]
78
79 if idiomas_filtrados:
80     juego["idiomas"] = idiomas_filtrados
81 elif "idiomas" in juego:
82     del juego["idiomas"]
83
84 # Agregar idiomas al formato deseado a cada juego
85 cont = 1
86 for juego in juegos_json:
87     print(cont)
88     cont += 1
89     agregar_idiomas_a_juego(juego)
90
91 # Guardar el resultado en un nuevo archivo JSON
92 with open("../data_02/games_with_languages.json", "w") as json_file:
93     json.dump(juegos_json, json_file, indent=4)
94
95 print("Archivo JSON 'games_with_languages.json' creado con éxito.")
96
```

Luego de todo el proceso mencionado anteriormente, se tiene la producción del json de juegos, el cual fué importado a MongoDB en el siguiente formato:

```
{ } games_with_alljson X
Proyecto2 > data_02 > { } games_with_alljson
1  [{
2    "_id": {
3      "$oid": "652d670b6602666692c62e3b"
4    },
5    "tipo": "Main Game",
6    "nombre": "Thief",
7    "igdb_id": 4,
8    "calificacion_general": 70.56635450852384,
9    "fecha_lanzamiento_general": "2014-02-24",
10   "critic_rating": 17,
11   "calificacion_profesional": 63.64285714285715,
12   "member_ratings": 269,
13   "total_calificaciones": 286,
14   "serie": "Thief",
15   "motores": [
16     {
17       "nombre": "Unreal Engine 3"
18     }
19   ],
20   "empresas": [
21     {
22       "nombre": "Eidos Montréal"
23     },
24     {
25       "nombre": "Square Enix"
26     }
27   ],
28   "generos": [
29     {
30       "tipo_genero": "Shooter"
31     },
32     {
33       "tipo_genero": "Adventure"
34     }
35   ],
36   "temas": [
37     {
38       "tipo": "Sandbox"
39     },
40     {
41       "tipo": "Action"
42     },
43     {
44       "tipo": "Stealth"
45     }
46   ],
47   "perspectivas": [
48     {
49       "tipo_perspectiva": "First person"
50     }
51   ],
52   "titulos": [
53     {
54       "nombre": "Thief IV",
55       "comentario": "Other"
56     },
57     {
58       "nombre": "Thief 4",
59       "comentario": "Other"
60     }
61   ]
62 }
```

- Colección de géneros

Esta información ya se encontraba en los archivos **.csv**, por lo que únicamente se procedió al procesamiento de datos y su respectiva conversión a **.json** con el siguiente código:

```
genres_game.py X
Proyecto2 > genres_game.py > ...
1 import pandas as pd
2 import json
3 import math
4
5 # Leer el archivo de géneros
6 generos_df = pd.read_csv('./data/generos.csv')
7 generos_dict = dict(zip(generos_df['id'], generos_df['tipo_genero']))
8
9 # Leer el archivo de relaciones entre videojuegos y géneros
10 videojuego_genero_df = pd.read_csv('./data/Videojuego_Genero.csv')
11
12 # Leer el archivo de juegos sin párrafos
13 juegos_df = pd.read_csv('./data/juegos_sin_parrafos.csv')
14
15 # Leer el archivo de relaciones entre videojuegos y plataformas
16 videojuego_plataforma_df = pd.read_csv('./data/Videojuego_Plataforma.csv')
17
18 # Leer el archivo de plataformas
19 plataformas_df = pd.read_csv('./data/plataformas.csv')
20 plataformas_dict = dict(zip(plataformas_df['plataforma_id'], plataformas_df['plataforma_nombre']))
21
22 # Crear un diccionario para mapear el IGDB_ID a la información del juego
23 juego_info_dict = {}
24 for _, row in juegos_df.iterrows():
25     igdb_id = row['igdb_id']
26     nombre = row['nombre']
27     calificacion_general = row['calificacion_general']
28     calificacion_profesional = row['calificacion_profesional']
29
30     # Validar si Los valores son NaN y reemplazarlos con None
31     if math.isnan(calificacion_general):
32         calificacion_general = None
33     if math.isnan(calificacion_profesional):
34         calificacion_profesional = None
35
36     juego_info_dict[int(igdb_id)] = {
37         'nombre': nombre,
38         'calificacion_general': calificacion_general,
39         'calificacion_profesional': calificacion_profesional,
40         'plataformas': [] # Agregar un atributo para las plataformas
41     }
42
43 # Agregar plataformas a los juegos
44 for _, row in videojuego_plataforma_df.iterrows():
45     igdb_id = int(row['game_id'])
46     plataforma_id = int(row['platform_id'])
47     plataforma_nombre = plataformas_dict.get(plataforma_id)
48     if igdb_id in juego_info_dict and plataforma_nombre:
49         if plataforma_nombre not in juego_info_dict[igdb_id]['plataformas']:
50             juego_info_dict[igdb_id]['plataformas'].append(plataforma_nombre)
51
52 # Crear la lista de juegos para cada género
53 generos_conjuegos = {}
54 for _, row in videojuego_genero_df.iterrows():
55     igdb_genero_id = int(row['Genero_Id']) # Convertir a int
56     igdb_id = int(row['Videojuego_Id']) # Convertir a int
57     genero_nombre = generos_dict.get(igdb_genero_id, 'Desconocido')
58     juego_info = juego_info_dict.get(igdb_id)
59     if igdb_genero_id not in generos_conjuegos:
```

genres_game.py X

Proyecto2 > genres_game.py > ...

```
41     }
42
43     # Agregar plataformas a los juegos
44     for _, row in videojuego_plataforma_df.iterrows():
45         igdb_id = int(row['game_id'])
46         plataforma_id = int(row['platform_id'])
47         plataforma_nombre = plataformas_dict.get(plataforma_id)
48         if igdb_id in juego_info_dict and plataforma_nombre:
49             if plataforma_nombre not in juego_info_dict[igdb_id]['plataformas']:
50                 juego_info_dict[igdb_id]['plataformas'].append(plataforma_nombre)
51
52     # Crear la lista de juegos para cada género
53     generos_conjuegos = {}
54     for _, row in videojuego_genero_df.iterrows():
55         igdb_genero_id = int(row['Genero_Id']) # Convertir a int
56         igdb_id = int(row['Videojuego_Id']) # Convertir a int
57         genero_nombre = generos_dict.get(igdb_genero_id, 'Desconocido')
58         juego_info = juego_info_dict.get(igdb_id)
59         if igdb_genero_id not in generos_conjuegos:
60             generos_conjuegos[igdb_genero_id] = {
61                 'igdb_genero_id': igdb_genero_id,
62                 'nombre': genero_nombre,
63                 'juegos': []
64             }
65         if juego_info:
66             # Validar si los valores no son None y no agregarlos al diccionario si son None
67             if juego_info['calificacion_general'] is not None or juego_info['calificacion_profesional'] is not None:
68                 generos_conjuegos[igdb_genero_id]['juegos'].append({
69                     'igdb_id': igdb_id,
70                     'nombre': juego_info['nombre'],
71                     'calificacion_general': juego_info['calificacion_general'],
72                     'calificacion_profesional': juego_info['calificacion_profesional'],
73                     'plataformas': juego_info['plataformas'] # Agregar plataformas al juego
74                 })
75
76     # Quitar atributos con valores None
77     for genero in generos_conjuegos.values():
78         for juego in genero['juegos']:
79             juego_copy = juego.copy()
80             for key, value in juego_copy.items():
81                 if value is None:
82                     del juego_copy[key]
83
84     # Convertir el resultado en una lista de géneros
85     generos_result = list(generos_conjuegos.values())
86
87     # Convertir a JSON
88     json_result = json.dumps(generos_result, default=str, indent=4) # Usar default=str para manejar tipos no serializables
89
90     # Guardar el JSON en un archivo
91     with open('output_03.json', 'w') as file:
92         file.write(json_result)
93
94     # Imprimir un mensaje de confirmación
95     print('JSON guardado en output_04.json')
96
```


Luego de todo el proceso mencionado anteriormente, se tiene la producción del json de géneros, este proceso iniciado el 21 de Octubre de 2023 a las 2:40 pm y concluyendo con la carga el mismo día a las 8:27 pm, el cual fué importado a MongoDB en el siguiente formato:

```
output_03.json X
Proyecto2 > data_02 > {} output_03.json > ...
1  [
2    {
3      "igdb_genre_id": 13,
4      "nombre": "Simulator",
5      "juegos": [
6        {
7          "igdb_id": 107423,
8          "nombre": "Mad Tower Tycoon",
9          "calificacion_general": 70.0,
10         "plataformas": [
11           "Mac",
12           "Xbox One",
13           "PC (Microsoft Windows)",
14           "PlayStation 4",
15           "Nintendo Switch"
16         ]
17       },
18       {
19         "igdb_id": 29049,
20         "nombre": "Drone Fighters",
21         "calificacion_general": 70.0,
22         "plataformas": [
23           "PC (Microsoft Windows)"
24         ]
25       },
26       {
27         "igdb_id": 35429,
28         "nombre": "The Lost Valley",
29         "calificacion_general": 40.0,
30         "plataformas": [
31           "PC (Microsoft Windows)"
32         ]
33       },
34       {
35         "igdb_id": 2667,
36         "nombre": "FlatOut",
37         "calificacion_general": 74.7148656221801,
38         "calificacion_profesional": 80.0,
39         "plataformas": [
40           "PlayStation 2",
41           "Xbox",
42           "PC (Microsoft Windows)",
43           "Linux"
44         ]
45       },
46       {
47         "igdb_id": 10355,
48         "nombre": "Gear City",
49         "calificacion_general": 80.0,
50         "plataformas": [
51           "Mac",
52           "PC (Microsoft Windows)",
53           "Linux"
54         ]
55       },
56       {
57         "igdb_id": 7571,
58         "nombre": "Car Mechanic Simulator 2014",
59         "calificacion_general": 60.02014856131026
60       }
61     ]
62   ]
```