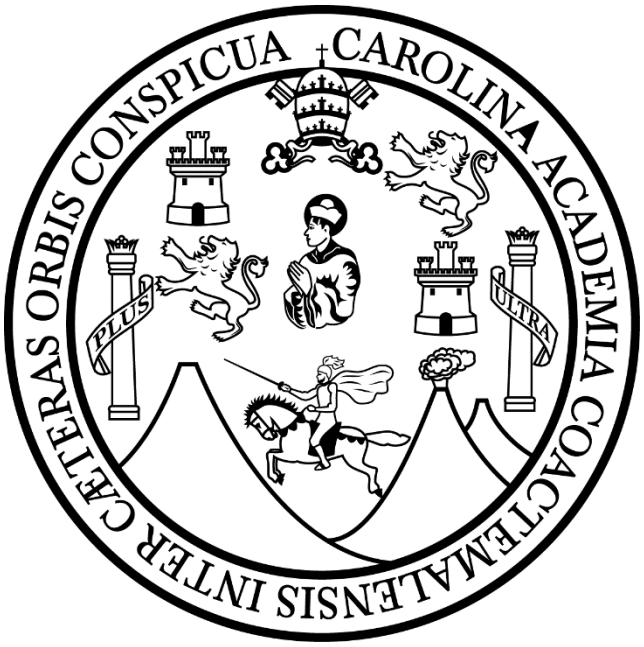
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

SISTEMAS DE BASES DE DATOS 2

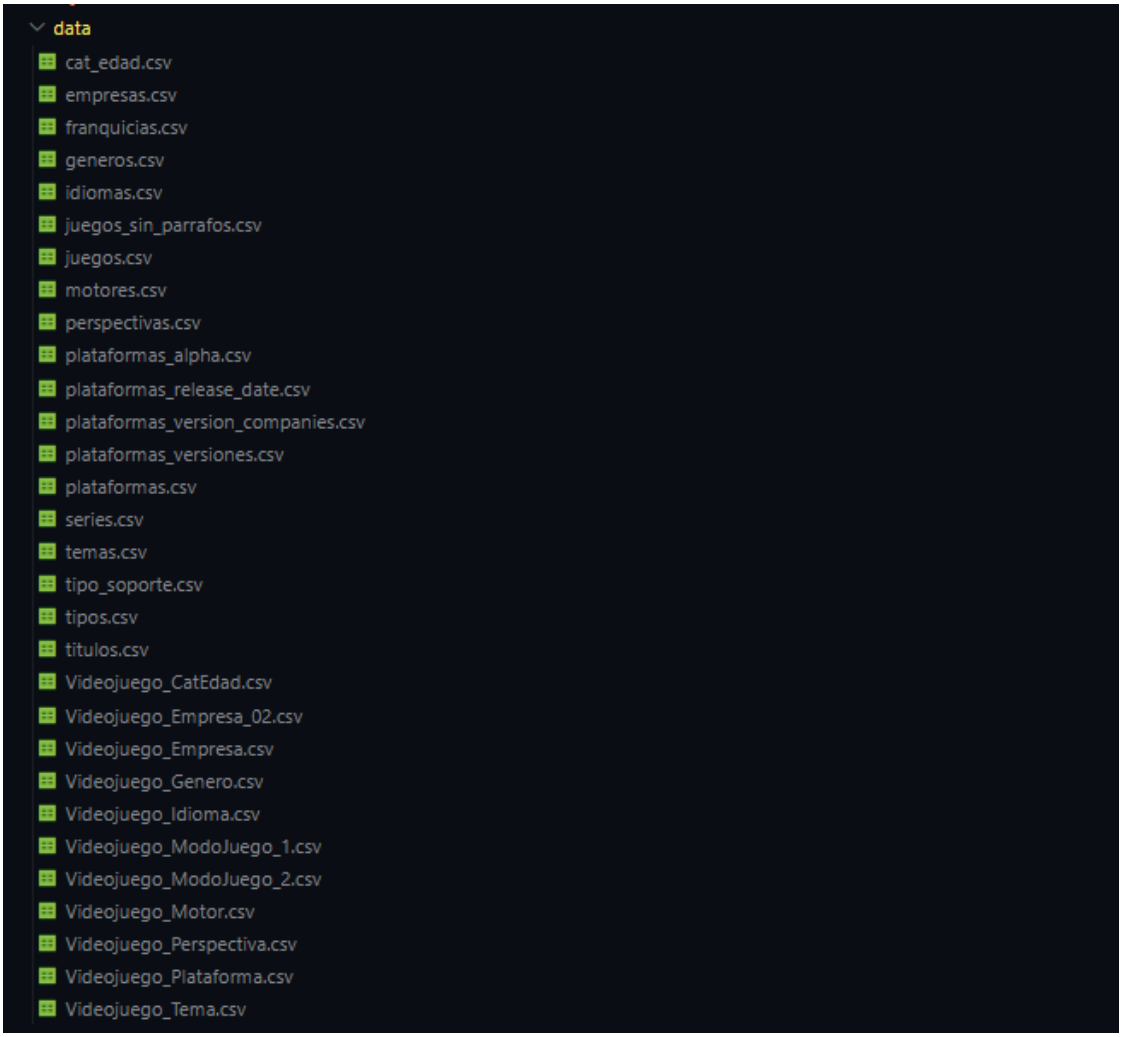
SECCIÓN A



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.**

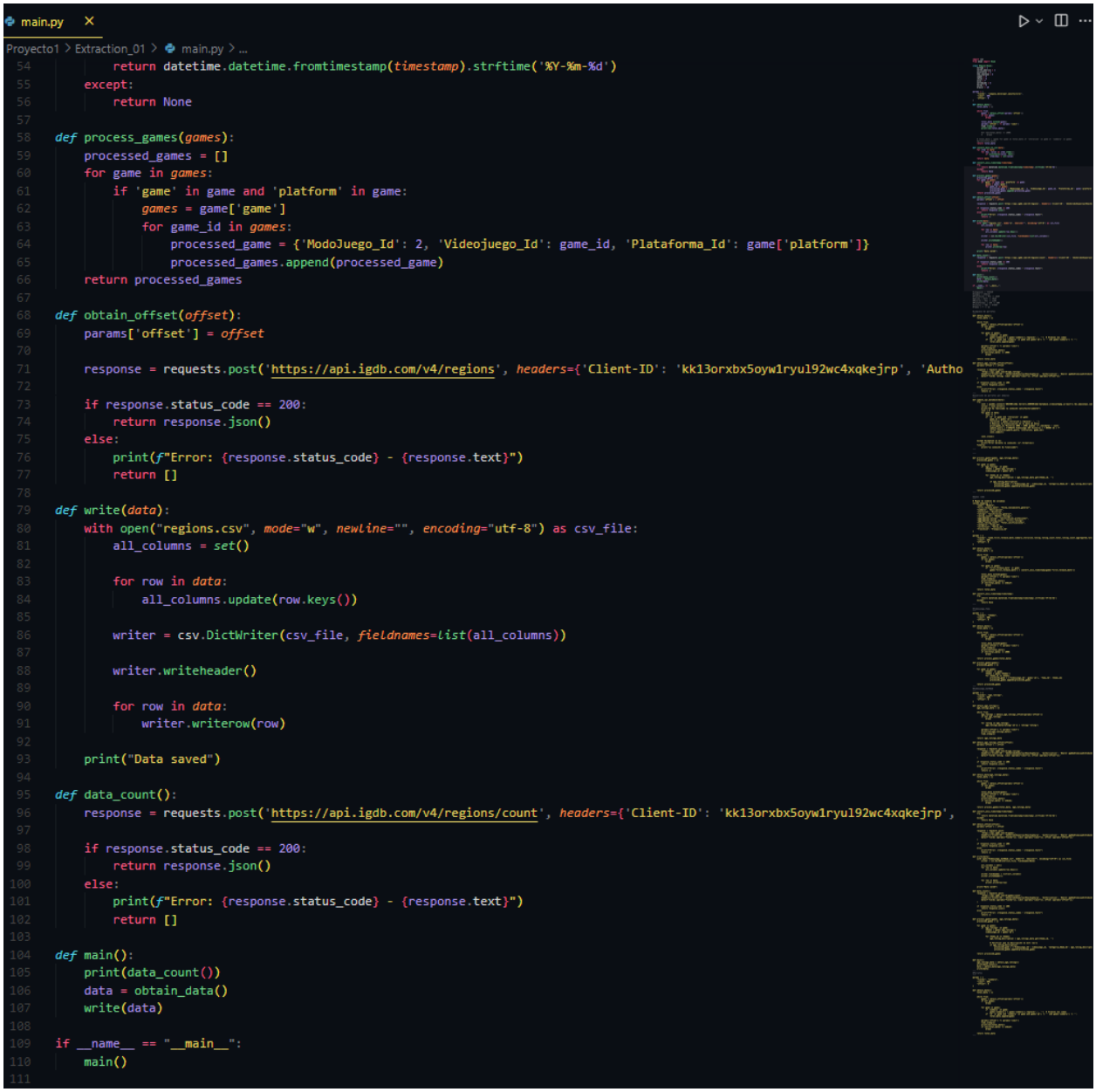


También se solicitó como requerimiento la obtención específica de datos sobre plataformas, esta información al variar un poco con la información almacenada en el Proyecto 1 si fué necesaria la utilización de la API nuevamente para la obtención de estos nuevos datos, pero eso se detalla más adelante.

- **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.

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 extraida se realizó de la misma manera, únicamente cambiando el endpoint sobre el cual se deseaba consultar y los parámetros que se necesitaban.



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

Proyecto2 > data > plataformas_alpha.csv

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

- Plataformas_versiones

plataformas_versiones.csv

Proyecto2 > data > plataformas_versiones.csv

```
platform_version_release_dates,companies,main_manufacturer,id,summary,name
[552],[533],,103,"The Commodore PET is a line of home/personal computers produced starting
[359],[527],,282,"A smaller version of the Switch, which only supports handheld play. The J
[204],[144],,139,Original Japanese version (SHVC-001),Super Famicom (SHVC-001)
,,,164,,Initial version
,,,222,,Initial version
"[360, 361, 362]",[368],,283,"A smaller version of the Wii with a top-loading disc drive. T
"[36, 37, 39, 133]",,,51,"In the early 1990s there was a trend by video game companies to u

Chairman of Nintendo of America, Howard Lincoln, teamed with Jim Clark, Chairman of Silicon

On July 18, 1994, Nintendo revealed that the new name for their system, the ""Ultra 64"" (1

Nintendo announced that it would release the Nintendo 64 system to American consumers on Ap

When the Nintendo 64 system was finally launched on September 29, 1996, only two games were

The Nintendo 64 system and the game Super Mario 64 opened to rave reviews from the critics.

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

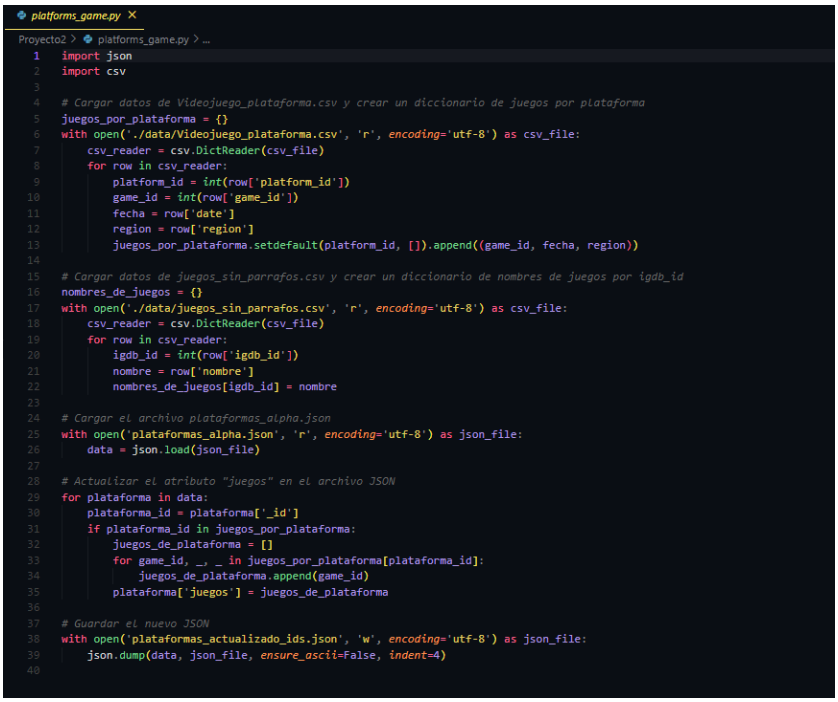
platforms.py

Proyecto2 > platforms.py > ...

```python
import csv
import json

# ENUM para tipo
tipo_enum = {
    "1": "console",
    "2": "arcade",
    "3": "platform",
    "4": "operating_system",
    "5": "portable_console",
    "6": "computer"
}

# ENUM para familia
familia_enum = {
    "5": "Nintendo",
    "4": "Linux",
    "2": "Xbox",
    "3": "Sega",
    "1": "PlayStation"
}

# Mapeo de regiones según el enum
region_enum = {
    "1": "europe",
    "2": "north_america",
    "3": "australia",
    "4": "new_zealand",
    "5": "japan",
    "6": "china",
    "7": "asia",
    "8": "worldwide",
    "9": "korea",
    "10": "brazil"
}

data = []

# Crea un diccionario para almacenar todos los datos del archivo "plataformas_alpha.csv"
csv_data = {}

# Abre el archivo "plataformas_alpha.csv" y lee los datos
with open("./data/plataformas_alpha.csv", mode="r", encoding="utf-8") as csv_file:
    csv_reader = csv.DictReader(csv_file)

    for row in csv_reader:
        csv_data[row["id"]] = row

# Crea un diccionario para almacenar todos los datos del archivo "plataformas_versiones.csv"
versions_data = {}

# Abre el archivo "plataformas_versiones.csv" y lee los datos
with open("./data/plataformas_versiones.csv", mode="r", encoding="utf-8") as versions_file:
    versions_reader = csv.DictReader(versions_file)

    for row in versions_reader:
        versions_data[row["id"]] = row

```

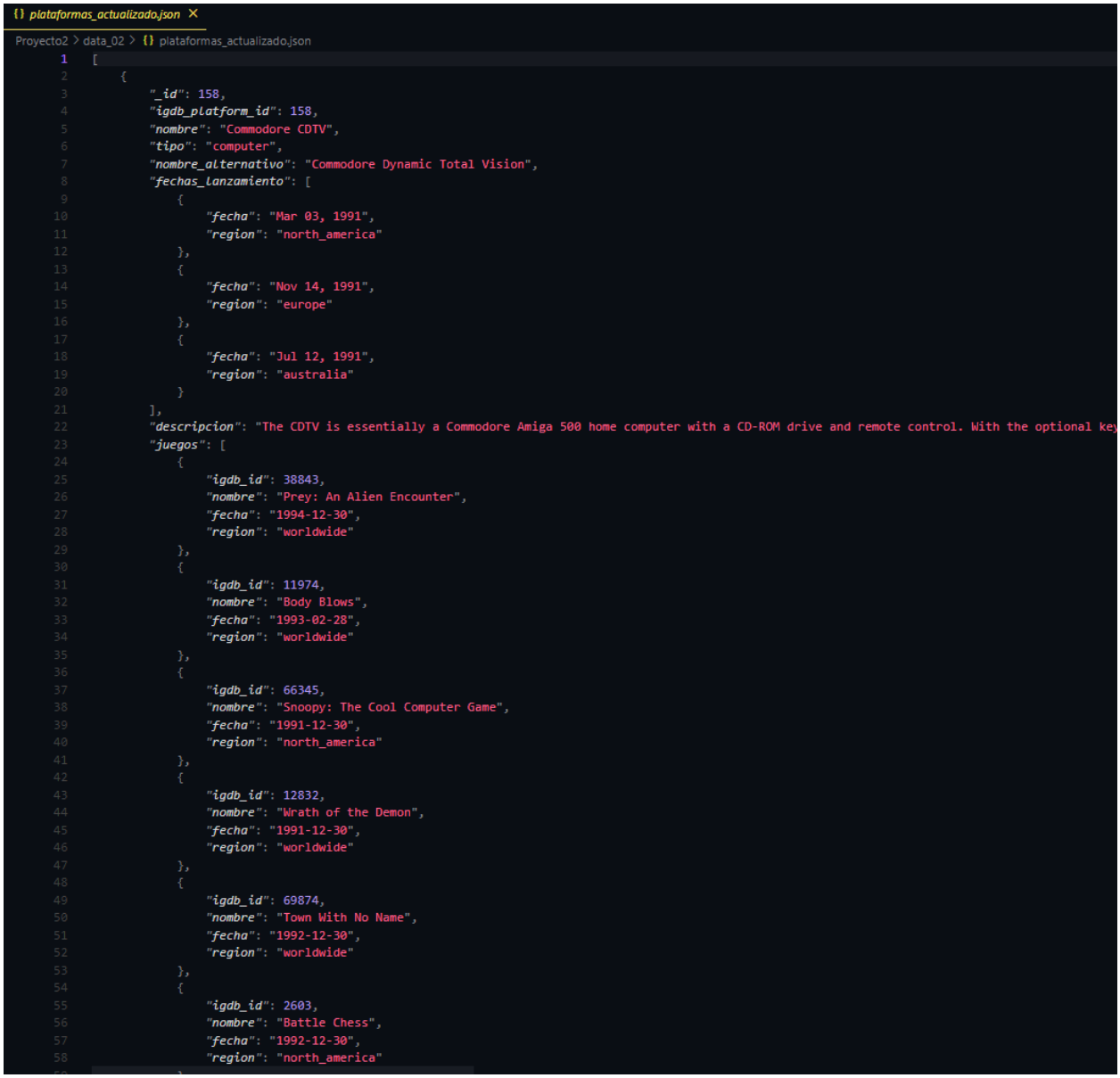
```python
    # Agrega nombre alternativo si está presente
    alternative_name = row.get("alternative_name")
    if alternative_name:
        json_data["nombre_alternativo"] = alternative_name

    # Procesa el primer elemento del array "versions" para buscar fechas de lanzamiento y descripción
    versions_str = row.get("versions")
    if versions_str:
        versions = json.loads(versions_str.replace("'", "\""))
        if versions and len(versions) > 0:
            version_id = versions[0]
            if str(version_id) in versions_data:
                platform_version = versions_data[str(version_id)]
                platform_version_release_dates_str = platform_version.get("platform_version_release_dates")
                if platform_version_release_dates_str:
                    release_date_ids = json.loads(platform_version_release_dates_str.replace("'", "\""))
                    if release_date_ids:
                        release_dates = []
                        for release_date_id in release_date_ids:
                            if str(release_date_id) in release_date_data:
                                release_date_info = release_date_data[str(release_date_id)]
                                region = region_enum.get(release_date_info["region"].lower(), release_date_info["region"])
                                release_dates.append({"fecha": release_date_info["fecha"], "region": region})
                        if release_dates:
                            json_data["fechas_lanzamiento"] = release_dates

                # Obtiene la descripción desde el primer elemento del array "versions"
                version_description = platform_version.get("summary")
                if version_description:
                    json_data["descripcion"] = version_description

    # Busca empresas asociadas
    version_companies_str = platform_version.get("companies")
    if version_companies_str:
        version_companies_ids = json.loads(version_companies_str.replace("'", "\""))
        if version_companies_ids:
            company_names = []
            for company_id in version_companies_ids:
                company_id = str(company_id)
                if company_id in version_companies_data:
                    company_id_in_companies = version_companies_data[company_id]["company"]
                    if company_id_in_companies in company_data:
                        company_names.append({"nombre": company_data[company_id_in_companies]["company_name"]})
            if company_names:
                json_data["empresas"] = company_names

    data.append(json_data)

# Escribe los datos en un archivo JSON
with open("plataformas_alpha.json", mode="w", encoding="utf-8") as json_file:
    json.dump(data, json_file, ensure_ascii=False, indent=4)

print("Datos convertidos y guardados en plataformas_alpha.json")
```

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



```python
import json
import csv

# Cargar datos de Videojuego_plataforma.csv y crear un diccionario de juegos por plataforma
juegos_por_plataforma = {}
with open('./data/Videojuego_plataforma.csv', 'r', encoding='utf-8') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    for row in csv_reader:
        platform_id = int(row['platform_id'])
        game_id = int(row['game_id'])
        fecha = row['date']
        region = row['region']
        juegos_por_plataforma.setdefault(platform_id, []).append((game_id, fecha, region))

# Cargar datos de juegos_sin_parrafos.csv y crear un diccionario de nombres de juegos por igdb_id
nombres_de_juegos = {}
with open('./data/juegos_sin_parrafos.csv', 'r', encoding='utf-8') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    for row in csv_reader:
        igdb_id = int(row['igdb_id'])
        nombre = row['nombre']
        nombres_de_juegos[igdb_id] = nombre

# Cargar el archivo plataformas_alpha.json
with open('plataformas_alpha.json', 'r', encoding='utf-8') as json_file:
    data = json.load(json_file)

# Actualizar el atributo "juegos" en el archivo JSON
for plataforma in data:
    plataforma_id = plataforma['_id']
    if plataforma_id in juegos_por_plataforma:
        juegos_de_plataforma = []
        for game_id, _, _ in juegos_por_plataforma[plataforma_id]:
            juegos_de_plataforma.append(game_id)
        plataforma['juegos'] = juegos_de_plataforma

# Guardar el nuevo JSON
with open('plataformas_actualizado_ids.json', 'w', encoding='utf-8') as json_file:
    json.dump(data, json_file, ensure_ascii=False, indent=4)
```

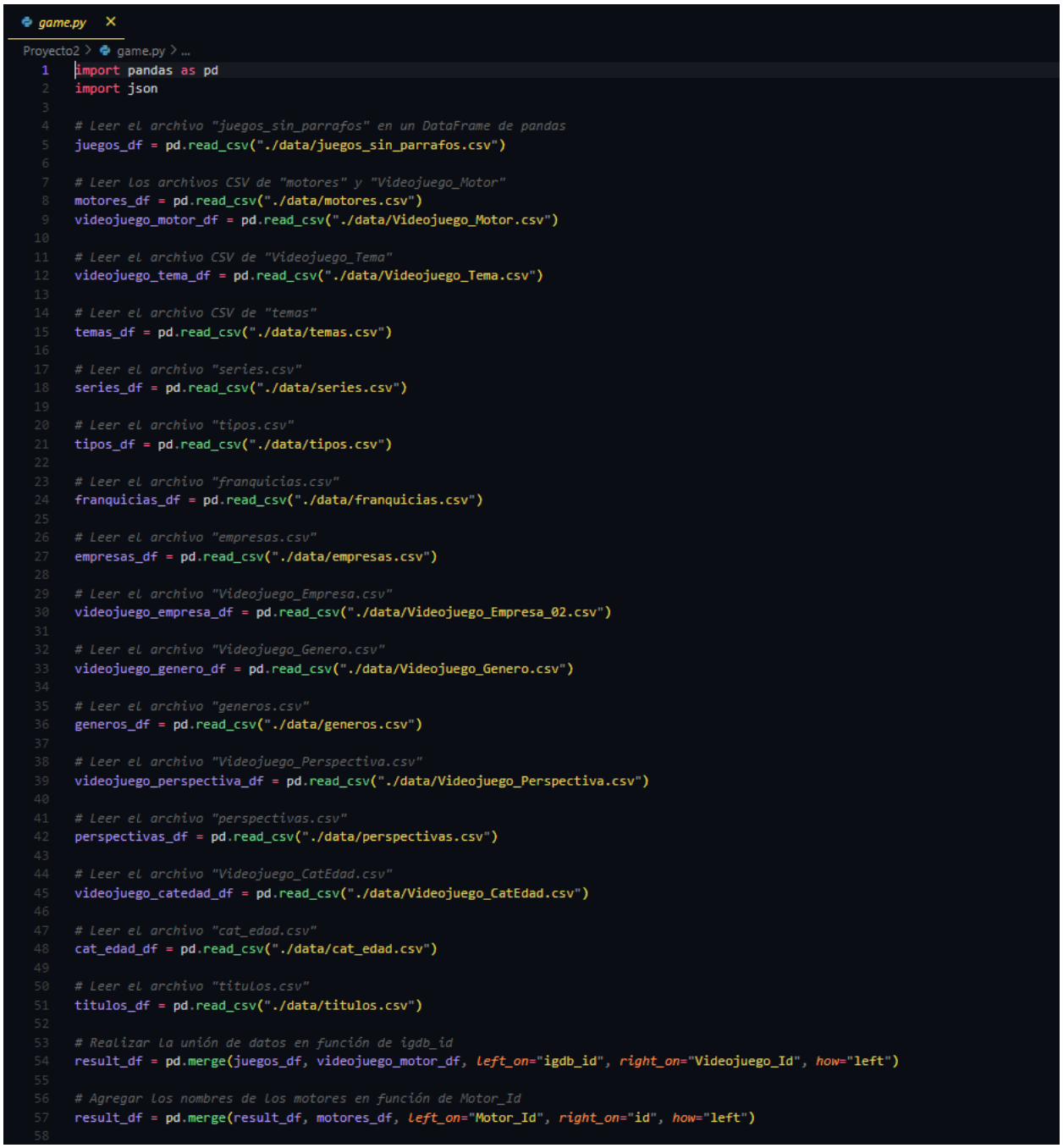
Luego de todo el proceso mencionado anteriormente, se tiene la producción del json con plataformas, el cual fué importado a Mongodb en el siguiente formato:




```
[
    {
        "_id": 158,
        "igdb_platform_id": 158,
        "nombre": "Commodore CDTV",
        "tipo": "computer",
        "nombre_alternativo": "Commodore Dynamic Total Vision",
        "fechas_lanzamiento": [
            {
                "fecha": "Mar 03, 1991",
                "region": "north_america"
            },
            {
                "fecha": "Nov 14, 1991",
                "region": "europe"
            },
            {
                "fecha": "Jul 12, 1991",
                "region": "australia"
            }
        ],
        "descripcion": "The CDTV is essentially a Commodore Amiga 500 home computer with a CD-ROM drive and remote control. With the optional key
        "juegos": [
            {
                "igdb_id": 38843,
                "nombre": "Prey: An Alien Encounter",
                "fecha": "1994-12-30",
                "region": "worldwide"
            },
            {
                "igdb_id": 11974,
                "nombre": "Body Blows",
                "fecha": "1993-02-28",
                "region": "worldwide"
            },
            {
                "igdb_id": 66345,
                "nombre": "Snoopy: The Cool Computer Game",
                "fecha": "1991-12-30",
                "region": "north_america"
            },
            {
                "igdb_id": 12832,
                "nombre": "Wrath of the Demon",
                "fecha": "1991-12-30",
                "region": "worldwide"
            },
            {
                "igdb_id": 69874,
                "nombre": "Town With No Name",
                "fecha": "1992-12-30",
                "region": "worldwide"
            },
            {
                "igdb_id": 2603,
                "nombre": "Battle Chess",
                "fecha": "1992-12-30",
                "region": "north_america"
```

- **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

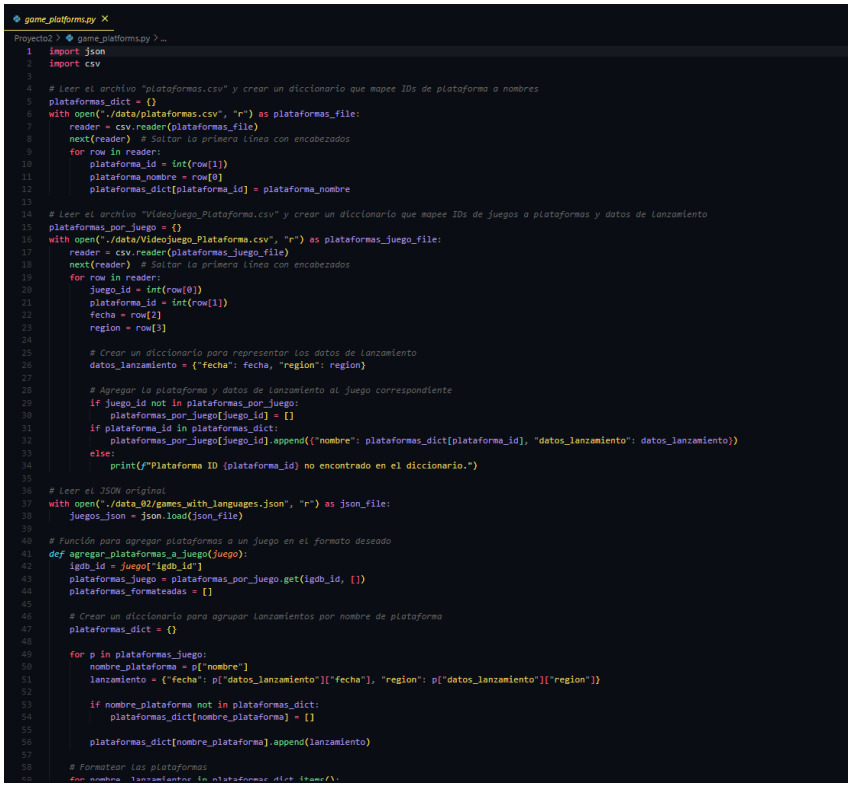
Proyecto2 > game.py > ...

```python
import pandas as pd
import json

# Leer el archivo "juegos_sin_parrafos" en un DataFrame de pandas
juegos_df = pd.read_csv("./data/juegos_sin_parrafos.csv")

# Leer los archivos CSV de "motores" y "Videojuego_Motor"
motores_df = pd.read_csv("./data/motores.csv")
videojuego_motor_df = pd.read_csv("./data/Videojuego_Motor.csv")

# Leer el archivo CSV de "Videojuego_Tema"
videojuego_tema_df = pd.read_csv("./data/Videojuego_Tema.csv")

# Leer el archivo CSV de "temas"
temas_df = pd.read_csv("./data/temas.csv")

# Leer el archivo "series.csv"
series_df = pd.read_csv("./data/series.csv")

# Leer el archivo "tipos.csv"
tipos_df = pd.read_csv("./data/tipos.csv")

# Leer el archivo "franquicias.csv"
franquicias_df = pd.read_csv("./data/franquicias.csv")

# Leer el archivo "empresas.csv"
empresas_df = pd.read_csv("./data/empresas.csv")

# Leer el archivo "Videojuego_Empresa.csv"
videojuego_empresa_df = pd.read_csv("./data/Videojuego_Empresa_02.csv")

# Leer el archivo "Videojuego_Genero.csv"
videojuego_genero_df = pd.read_csv("./data/Videojuego_Genero.csv")

# Leer el archivo "generos.csv"
generos_df = pd.read_csv("./data/generos.csv")

# Leer el archivo "Videojuego_Perspectiva.csv"
videojuego_perspectiva_df = pd.read_csv("./data/Videojuego_Perspectiva.csv")

# Leer el archivo "perspectivas.csv"
perspectivas_df = pd.read_csv("./data/perspectivas.csv")

# Leer el archivo "Videojuego_CatEdad.csv"
videojuego_catedad_df = pd.read_csv("./data/Videojuego_CatEdad.csv")

# Leer el archivo "cat_edad.csv"
cat_edad_df = pd.read_csv("./data/cat_edad.csv")

# Leer el archivo "titulos.csv"
titulos_df = pd.read_csv("./data/titulos.csv")

# Realizar la unión de datos en función de igdb_id
result_df = pd.merge(juegos_df, videojuego_motor_df, left_on="igdb_id", right_on="Videojuego_Id", how="left")

# Agregar los nombres de los motores en función de Motor_Id
result_df = pd.merge(result_df, motores_df, left_on="Motor_Id", right_on="id", how="left")

```

```python
        documento["motores"] = motores

    # Agregar empresas sin duplicados
    empresas = [{"nombre": nombre} for nombre in set(group["company_name"].dropna().values)]
    if empresas:
        documento["empresas"] = empresas

    # Agregar géneros sin duplicados
    generos = [{"tipo_genero": tipo} for tipo in set(group["tipo_genero"].dropna().values)]
    if generos:
        documento["generos"] = generos

    # Agregar temas sin duplicados
    temas = [{"tipo": tipo} for tipo in set(group["name_y"].dropna().values)]
    if temas:
        documento["temas"] = temas

    # Agregar perspectivas sin duplicados
    perspectivas = [{"tipo_perspectiva": tipo} for tipo in set(group["tipo_perspectiva"].dropna().values)]
    if perspectivas:
        documento["perspectivas"] = perspectivas

    # Agregar categorías de edad sin duplicados
    catedades = [{"tipo_cat_edad": tipo} for tipo in set(group["tipo_cat_edad"].dropna().values)]
    if catedades:
        documento["categorias_edad"] = catedades

    # Agregar títulos con comentarios sin duplicados y sin valores NaN
    titulos = []
    titulos_set = set()  # Conjunto para rastrear títulos duplicados
    for _, row in group.iterrows():
        titulo_nombre = row["title"]
        comentario = row["comment"]
        if not pd.isna(titulo_nombre) or not pd.isna(comentario):
            if titulo_nombre not in titulos_set:
                titulo = {}
                if not pd.isna(titulo_nombre):
                    titulo["nombre"] = titulo_nombre
                if not pd.isna(comentario):
                    titulo["comentario"] = comentario
                titulos.append(titulo)
                titulos_set.add(titulo_nombre)
    if titulos:
        documento["titulos"] = titulos

    documentos.append(documento)

# Guardar el resultado en un archivo JSON final
with open("juegos_con_franquicias.json", "w") as json_file:
    json.dump(documentos, json_file, indent=4)

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.



```python
import json
import csv

# Leer el archivo "plataformas.csv" y crear un diccionario que mapee IDs de plataforma a nombres
plataformas_dict = {}
with open('./data/plataformas.csv', "r") as plataformas_file:
    reader = csv.reader(plataformas_file)
    next(reader)  # Saltar la primera línea con encabezados
    for row in reader:
        plataforma_id = int(row[1])
        plataforma_nombre = row[0]
        plataformas_dict[plataforma_id] = plataforma_nombre

# Leer el archivo "Videojuego_Plataforma.csv" y crear un diccionario que mapee IDs de juegos a plataformas y datos de lanzamiento
plataformas_por_juego = {}
with open('./data/Videojuego_Plataforma.csv', "r") as plataformas_juego_file:
    reader = csv.reader(plataformas_juego_file)
    next(reader)  # Saltar la primera línea con encabezados
    for row in reader:
        juego_id = int(row[0])
        plataforma_id = int(row[1])
        fecha = row[2]
        region = row[3]

        # Crear un diccionario para representar los datos de lanzamiento
        datos_lanzamiento = {"fecha": fecha, "region": region}

        # Agregar la plataforma y datos de lanzamiento al juego correspondiente
        if juego_id not in plataformas_por_juego:
            plataformas_por_juego[juego_id] = []
        if plataforma_id in plataformas_dict:
            plataformas_por_juego[juego_id].append({"nombre": plataformas_dict[plataforma_id], "datos_lanzamiento": datos_lanzamiento})
        else:
            print(f"Plataforma ID {plataforma_id} no encontrado en el diccionario.")

# Leer el JSON original
with open("./data_02/games_with_languages.json", "r") as json_file:
    juegos_json = json.load(json_file)

# Función para agregar plataformas a un juego en el formato deseado
def agregar_plataformas_a_juego(juego):
    igdb_id = juego["igdb_id"]
    plataformas_juego = plataformas_por_juego.get(igdb_id, [])
    plataformas_formateadas = []

    # Crear un diccionario para agrupar lanzamientos por nombre de plataforma
    plataformas_dict = {}

    for p in plataformas_juego:
        nombre_plataforma = p["nombre"]
        lanzamiento = {"fecha": p["datos_lanzamiento"]["fecha"], "region": p["datos_lanzamiento"]["region"]}

        if nombre_plataforma not in plataformas_dict:
            plataformas_dict[nombre_plataforma] = []

        plataformas_dict[nombre_plataforma].append(lanzamiento)

    # Formatear las plataformas
    for nombre, lanzamientos in plataformas_dict.items():
```

```python
with open("./data_02/games_with_languages.json", "r") as json_file:
    juegos_json = json.load(json_file)

# Función para agregar plataformas a un juego en el formato deseado
def agregar_plataformas_a_juego(juego):
    igdb_id = juego["igdb_id"]
    plataformas_juego = plataformas_por_juego.get(igdb_id, [])
    plataformas_formateadas = []

    # Crear un diccionario para agrupar lanzamientos por nombre de plataforma
    plataformas_dict = {}

    for p in plataformas_juego:
        nombre_plataforma = p["nombre"]
        lanzamiento = {"fecha": p["datos_lanzamiento"]["fecha"], "region": p["datos_lanzamiento"]["region"]}

        if nombre_plataforma not in plataformas_dict:
            plataformas_dict[nombre_plataforma] = []

        plataformas_dict[nombre_plataforma].append(lanzamiento)

    # Formatear las plataformas
    for nombre, lanzamientos in plataformas_dict.items():
        plataforma_formato = {"nombre": nombre, "lanzamientos": lanzamientos}
        plataformas_formateadas.append(plataforma_formato)

    # Validación: No agregar "plataformas" si el array está vacío
    if plataformas_formateadas:
        juego["plataformas"] = plataformas_formateadas
    elif "plataformas" in juego:
        del juego["plataformas"]

# Agregar plataformas al formato deseado a cada juego
cont = 1
for juego in juegos_json:
    print(cont)
    cont += 1
    agregar_plataformas_a_juego(juego)

# Guardar el resultado en un nuevo archivo JSON
with open("./data_02/games_with_platforms.json", "w") as json_file:
    json.dump(juegos_json, json_file, indent=4)

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

Proyecto2 > game_languages.py > ...

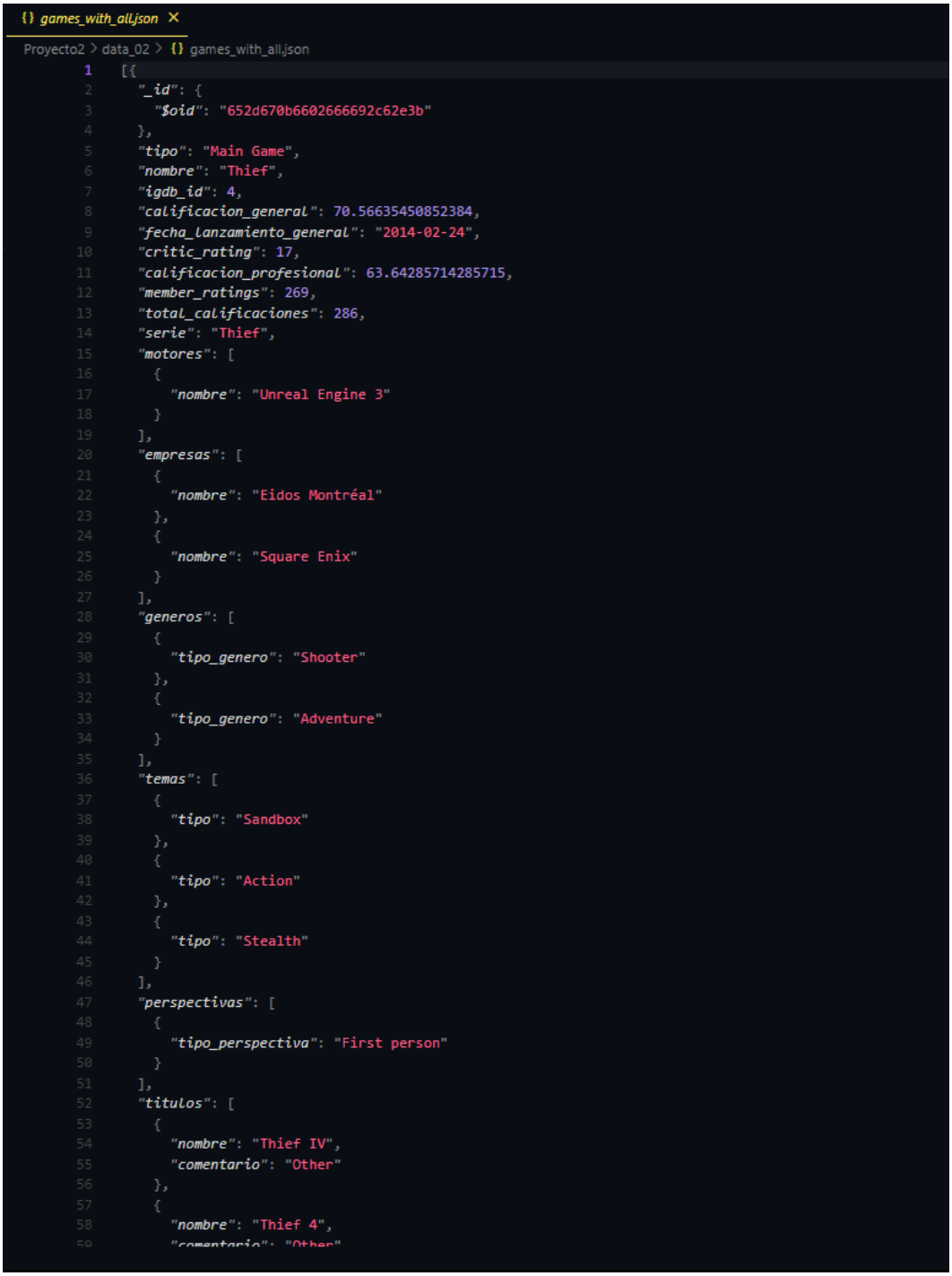
```python
import json

# Crear un diccionario con la información de idiomas
idiomas_dict = {
    1: "Arabic",
    2: "Chinese (Simplified)",
    3: "Chinese (Traditional)",
    4: "Czech",
    5: "Danish",
    6: "Dutch",
    7: "English",
    8: "English (UK)",
    9: "Spanish (Spain)",
    10: "Spanish (Mexico)",
    11: "Finnish",
    12: "French",
    13: "Hebrew",
    14: "Hungarian",
    15: "Italian",
    16: "Japanese",
    17: "Korean",
    18: "Norwegian",
    19: "Polish",
    20: "Portuguese (Portugal)",
    21: "Portuguese (Brazil)",
    22: "Russian",
    23: "Swedish",
    24: "Turkish",
    25: "Thai",
    26: "Vietnamese",
    27: "German",
    28: "Ukrainian"
}

# Crear un diccionario con la información de tipos de soporte
tipo_soporte_dict = {
    1: "Audio",
    2: "Subtitles",
    3: "Interface"
}

# Crear un diccionario para mapear IDs de juegos a listas de idiomas
idiomas_por_juego = {}

# Leer el archivo "Videojuego_Idioma.csv" y crear diccionarios que mapeen IDs a nombres
with open("./data/Videojuego_Idioma.csv", "r") as idiomas_file:
    next(idiomas_file)  # Saltar la primera línea con encabezados
    for line in idiomas_file:
        partes = line.strip().split(",")
        igdb_id = int(partes[1])
        id_idioma = int(partes[0])
        tipo_soporte_id = int(partes[2])
        if igdb_id not in idiomas_por_juego:
            idiomas_por_juego[igdb_id] = []
        idiomas_por_juego[igdb_id].append({"nombre": id_idioma, "tipo": tipo_soporte_id})

# Leer el JSON original
with open("./data_02/games.json", "r") as json_file:
    juegos_json = json.load(json_file)
```

game_languages.py

Proyecto2 > game_languages.py > ...

```python
            id_idioma = int(partes[0])
            tipo_soporte_id = int(partes[2])
            if igdb_id not in idiomas_por_juego:
                idiomas_por_juego[igdb_id] = []
            idiomas_por_juego[igdb_id].append({"nombre": id_idioma, "tipo": tipo_soporte_id})

# Leer el JSON original
with open("./data_02/games.json", "r") as json_file:
    juegos_json = json.load(json_file)

# Función para agregar idiomas a un juego en el formato deseado
def agregar_idiomas_a_juego(juego):
    igdb_id = juego["igdb_id"]
    idiomas_juego = idiomas_por_juego.get(igdb_id, [])
    idiomas_formateados = {}

    for idioma in idiomas_juego:
        nombre_idioma = idiomas_dict.get(idioma["nombre"], "Desconocido")
        tipo_soporte = tipo_soporte_dict.get(idioma["tipo"], "Desconocido")

        if nombre_idioma not in idiomas_formateados:
            idiomas_formateados[nombre_idioma] = {"nombre": nombre_idioma, "implementaciones": []}

        idiomas_formateados[nombre_idioma]["implementaciones"].append({"tipo": tipo_soporte})

    # Filtra y elimina los idiomas con arrays vacíos
    idiomas_filtrados = [idioma for idioma in idiomas_formateados.values() if idioma["implementaciones"]]

    if idiomas_filtrados:
        juego["idiomas"] = idiomas_filtrados
    elif "idiomas" in juego:
        del juego["idiomas"]

# Agregar idiomas al formato deseado a cada juego
cont = 1
for juego in juegos_json:
    print(cont)
    cont += 1
    agregar_idiomas_a_juego(juego)

# Guardar el resultado en un nuevo archivo JSON
with open("./data_02/games_with_languages.json", "w") as json_file:
    json.dump(juegos_json, json_file, indent=4)

print("Archivo JSON 'games_with_languages.json' creado con éxito.")

```

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_all.json

Proyecto2 > data_02 > games_with_all.json


```
[{
  "_id": {
    "$oid": "652d670b6602666692c62e3b"
  },
  "tipo": "Main Game",
  "nombre": "Thief",
  "igdb_id": 4,
  "calificacion_general": 70.56635450852384,
  "fecha_lanzamiento_general": "2014-02-24",
  "critic_rating": 17,
  "calificacion_profesional": 63.64285714285715,
  "member_ratings": 269,
  "total_calificaciones": 286,
  "serie": "Thief",
  "motores": [
    {
      "nombre": "Unreal Engine 3"
    }
  ],
  "empresas": [
    {
      "nombre": "Eidos Montréal"
    },
    {
      "nombre": "Square Enix"
    }
  ],
  "generos": [
    {
      "tipo_genero": "Shooter"
    },
    {
      "tipo_genero": "Adventure"
    }
  ],
  "temas": [
    {
      "tipo": "Sandbox"
    },
    {
      "tipo": "Action"
    },
    {
      "tipo": "Stealth"
    }
  ],
  "perspectivas": [
    {
      "tipo_perspectiva": "First person"
    }
  ],
  "titulos": [
    {
      "nombre": "Thief IV",
      "comentario": "Other"
    },
    {
      "nombre": "Thief 4",
      "comentario": "Other"
```

- **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:

```python
import pandas as pd
import json
import math

# Leer el archivo de géneros
generos_df = pd.read_csv('./data/generos.csv')
generos_dict = dict(zip(generos_df['id'], generos_df['tipo_genero']))

# Leer el archivo de relaciones entre videojuegos y géneros
videojuego_genero_df = pd.read_csv('./data/Videojuego_Genero.csv')

# Leer el archivo de juegos sin párrafos
juegos_df = pd.read_csv('./data/juegos_sin_parrafos.csv')

# Leer el archivo de relaciones entre videojuegos y plataformas
videojuego_plataforma_df = pd.read_csv('./data/Videojuego_Plataforma.csv')

# Leer el archivo de plataformas
plataformas_df = pd.read_csv('./data/plataformas.csv')
plataformas_dict = dict(zip(plataformas_df['plataforma_id'], plataformas_df['plataforma_name']))

# Crear un diccionario para mapear el IGDB_ID a la información del juego
juego_info_dict = {}
for _, row in juegos_df.iterrows():
    igdb_id = row['igdb_id']
    nombre = row['nombre']
    calificacion_general = row['calificacion_general']
    calificacion_profesional = row['calificacion_profesional']

    # Validar si los valores son NaN y reemplazarlos con None
    if math.isnan(calificacion_general):
        calificacion_general = None
    if math.isnan(calificacion_profesional):
        calificacion_profesional = None

    juego_info_dict[int(igdb_id)] = {
        'nombre': nombre,
        'calificacion_general': calificacion_general,
        'calificacion_profesional': calificacion_profesional,
        'plataformas': []  # Agregar un atributo para las plataformas
    }

# Agregar plataformas a los juegos
for _, row in videojuego_plataforma_df.iterrows():
    igdb_id = int(row['game_id'])
    plataforma_id = int(row['platform_id'])
    plataforma_nombre = plataformas_dict.get(plataforma_id)
    if igdb_id in juego_info_dict and plataforma_nombre:
        if plataforma_nombre not in juego_info_dict[igdb_id]['plataformas']:
            juego_info_dict[igdb_id]['plataformas'].append(plataforma_nombre)

# Crear la lista de juegos para cada género
generos_conjuegos = {}
for _, row in videojuego_genero_df.iterrows():
    igdb_genre_id = int(row['Genero_Id'])  # Convertir a int
    igdb_id = int(row['Videojuego_Id'])  # Convertir a int
    genero_nombre = generos_dict.get(igdb_genre_id, 'Desconocido')
    juego_info = juego_info_dict.get(igdb_id)
```

genres_game.py

Proyecto2 > genres_game.py > ...

```python
    }

# Agregar plataformas a los juegos
for _, row in videojuego_plataforma_df.iterrows():
    igdb_id = int(row['game_id'])
    plataforma_id = int(row['platform_id'])
    plataforma_nombre = plataformas_dict.get(plataforma_id)
    if igdb_id in juego_info_dict and plataforma_nombre:
        if plataforma_nombre not in juego_info_dict[igdb_id]['plataformas']:
            juego_info_dict[igdb_id]['plataformas'].append(plataforma_nombre)

# Crear la lista de juegos para cada género
generos_conjuegos = {}
for _, row in videojuego_genero_df.iterrows():
    igdb_genre_id = int(row['Genero_Id'])  # Convertir a int
    igdb_id = int(row['Videojuego_Id'])  # Convertir a int
    genero_nombre = generos_dict.get(igdb_genre_id, 'Desconocido')
    juego_info = juego_info_dict.get(igdb_id)
    if igdb_genre_id not in generos_conjuegos:
        generos_conjuegos[igdb_genre_id] = {
            'igdb_genre_id': igdb_genre_id,
            'nombre': genero_nombre,
            'juegos': []
        }
    if juego_info:
        # Validar si los valores no son None y no agregarlos al diccionario si son None
        if juego_info['calificacion_general'] is not None or juego_info['calificacion_profesional'] is not None:
            generos_conjuegos[igdb_genre_id]['juegos'].append({
                'igdb_id': igdb_id,
                'nombre': juego_info['nombre'],
                'calificacion_general': juego_info['calificacion_general'],
                'calificacion_profesional': juego_info['calificacion_profesional'],
                'plataformas': juego_info['plataformas']  # Agregar plataformas al juego
            })

# Quitar atributos con valores None
for genero in generos_conjuegos.values():
    for juego in genero['juegos']:
        juego_copy = juego.copy()
        for key, value in juego_copy.items():
            if value is None:
                del juego[key]

# Convertir el resultado en una lista de géneros
generos_result = list(generos_conjuegos.values())

# Convertir a JSON
json_result = json.dumps(generos_result, default=str, indent=4)  # Usar default=str para manejar tipos no serializa

# Guardar el JSON en un archivo
with open('output_03.json', 'w') as file:
    file.write(json_result)

# Imprimir un mensaje de confirmación
print('JSON guardado en output_04.json')

```

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 U

Proyecto2 > data_02 > output_03.json > ...


```
[
    {
        "igdb_genre_id": 13,
        "nombre": "Simulator",
        "juegos": [
            {
                "igdb_id": 107423,
                "nombre": "Mad Tower Tycoon",
                "calificacion_general": 70.0,
                "plataformas": [
                    "Mac",
                    "Xbox One",
                    "PC (Microsoft Windows)",
                    "PlayStation 4",
                    "Nintendo Switch"
                ]
            },
            {
                "igdb_id": 29049,
                "nombre": "Drone Fighters",
                "calificacion_general": 70.0,
                "plataformas": [
                    "PC (Microsoft Windows)"
                ]
            },
            {
                "igdb_id": 35429,
                "nombre": "The Lost Valley",
                "calificacion_general": 40.0,
                "plataformas": [
                    "PC (Microsoft Windows)"
                ]
            },
            {
                "igdb_id": 2667,
                "nombre": "FlatOut",
                "calificacion_general": 74.7148656221801,
                "calificacion_profesional": 80.0,
                "plataformas": [
                    "PlayStation 2",
                    "Xbox",
                    "PC (Microsoft Windows)",
                    "Linux"
                ]
            },
            {
                "igdb_id": 10355,
                "nombre": "Gear City",
                "calificacion_general": 80.0,
                "plataformas": [
                    "Mac",
                    "PC (Microsoft Windows)",
                    "Linux"
                ]
            },
            {
                "igdb_id": 7571,
                "nombre": "Car Mechanic Simulator 2014",
                "calificacion_general": 69.92914856131926
```