Preparación de Datos para Consumo de API

Para que el cliente pueda consumir la API proporcionada correctamente, es necesario priorizar la velocidad de las solicitudes y la calidad de los datos de respuestas minimizando errores de tokenización.
Es por ello, que en este documento se realizaron las transformaciones necesarias para que la API fuera lo más óptima posible para los endpoints solicitados:

- **Género**: Se ingresa un año y devuelve una lista con los 5 géneros más ofrecidos en el orden correspondiente.
- **Juegos**: Se ingresa un año y devuelve una lista con los juegos lanzados en el año.
- **Specs**: Se ingresa un año y devuelve una lista con los 5 specs que más se repiten en el mismo en el orden correspondiente.
- **Early Access**: Cantidad de juegos lanzados en un año con early access.
- **Sentiment**: Según el año de lanzamiento, se devuelve una lista con la cantidad de registros que se encuentren categorizados con un análisis de sentimiento.
- **Metascore**: Top 5 juegos según año con mayor metascore.

Cabe resaltar que, de acuerdo a las características de estos endpoints, el año de lanzamiento es la columna más importante y con la cual se tomó especial atención.

Carga de Datos y Librerías

```
1 import pandas as pd
2 import pickle
3
4 df = pd.read_csv('/content/drive/MyDrive/datasets/steam_games_PI.csv', index_col=[0])
5 df.info()
```

```
Int64Index: 32135 entries, 0 to 32134
Data columns (total 16 columns):
 # Column Non-Null Count Dtype
--- ------ -------------- -----
 0 publisher 24073 non-null object
 1 genres 28852 non-null object
 2 app_name 32133 non-null object
 3 title 30085 non-null object
 4 url 32135 non-null object
 5 release_date 30068 non-null object
 6 tags 31972 non-null object
 7 discount_price 225 non-null float64
 8 reviews_url 32133 non-null object
 9 specs 31465 non-null object
 10 price 30758 non-null object
 11 early_access 32135 non-null bool
 12 id 32133 non-null float64
 13 developer 28836 non-null object
 14 sentiment 24953 non-null object
 15 metascore 2607 non-null float64
```

Datos nulos

Los datos nulos pueden generar problemas de tokenización al ser recibidos como JSON por parte del cliente. Por lo tanto, todos estos se trataron tomando diferentes consideraciones dependiendo del valor de los datos.

```
1 df.isnull().sum()
    publisher 8062
    genres 3283
    app_name 2
    title 2050
    url 0
    release_date 2067
    tags 163
    discount_price 31910
    reviews_url 2
    specs 670
    price 1377
    early_access 0
    id 2
    developer 3299
    sentiment 7182
    metascore 29528
```

Ciertas columnas se consideraron indispensables para que un videojuego pudiera ser un registro válido de la API. Para estos casos, la ausencia del valor se trató eliminando por completo el registro. Estos campos fueron:

- `id`
- `app_name`

Por otro lado, todos los endpoints solicitados hacen uso de la fecha, por lo que un videojuego sin fecha es un dato que nunca podría ser consultado.

```
1 df = df[df['app_name'].notna()]
2 df = df[df['id'].notna()]
3 df = df[df['release_date'].notna()]
```

Para “`discount_price`” y “`price`” los nulos se imputaron con ceros y en el resto de columnas, se agregó un mensaje indicando la ausencia de los datos.

```
 1 # Mensajes de Ausencia
 2 df['developer'] = df['developer'].fillna('No_developer_registered')
 3 df['tags'] = df['tags'].fillna('tags')
 4 df['publisher'] = df['publisher'].fillna('no_publisher_registered')
 5 df['genres'] = df['genres'].fillna('no_genres_registered')
 6 df['sentiment'] = df['sentiment'].fillna('no_sentiment')
 7 df['metascore'] = df['metascore'].fillna('no_score')
 8 df['specs'] = df['specs'].fillna('no_specs')
 9
10 # Imputación con Ceros
11 df['discount_price'] = df['discount_price'].fillna(0)
12 df['price'] = df['price'].fillna(0)
```

Fechas

Como se indicó antes, todos los endpoints solicitados hacen uso únicamente del año de lanzamiento de los videojuegos, por lo que se revisó que todas las fechas tuvieran el formato correcto: año-mes-día. De no serlo, se eliminó el registro.

```
df['release_date'] = pd.to_datetime(df['release_date'], format='%Y-%m-%d', errors='coerce')
print('Número de registros eliminados:', df['release_date'].isna().sum())
df = df.dropna(subset=['release_date'])
```

```
Número de valores eliminados: 243
```

Columnas: App_name y Title

Para las columnas `app_name` y `title` , se consideró que el significado propuesto por el diccionario de datos era muy similar para ambas columnas, por lo que se evaluó si ambas columnas contenían los mismos datos.

```
count = (df['title'] == df['app_name']).sum()
print(count, 'valores de', df.shape[0], 'son iguales,', df.shape[0] - count, 'son diferentes')
```

```
29269 valores de 29823 son iguales, 554 son diferentes
```

De acuerdo a esto, ambas columnas eran redundantes entre sí teniendo solo 554 registros diferentes. En este sentido, una de estas se eliminó del dataframe: `title` , porque a diferencia de `app_name` , esta tuvo valores nulos al comienzo del análisis.

```
df = df.drop("title", axis=1)
```

Exportar Datos Serializados

En este punto, los datos ya estaban listos para ser consumidos por la API. Se exportaron en formato pkl para reducir el peso del archivo y mejorar la velocidad de las consultas.

```
pkl_file = open('games_steam.pkl', 'ab')
pickle.dump(df, pkl_file)
pkl_file.close()
```