Clustering, Clasificación y Regresión en Datos de Películas de Hollywood

1. Introducción

""" Este trabajo práctico aplica técnicas de clustering, clasificación y regresión a un conjunto de datos reales sobre películas de Hollywood. El objetivo es explorar estructuras latentes en los datos, identificar patrones mediante agrupamiento, y predecir variables de interés como puntuación crítica o presencia de premios. También se analiza la relación entre los grupos temáticos y el éxito galardonado. """

2. Carga y Exploración del Dataset

In [1]:
```
# Cargar las librerías necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA, NMF
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score, adjusted_rand_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn.metrics import classification_report, mean_squared_error, r2_score, mean_absolute_error
from sklearn.manifold import TSNE
from sklearn.datasets import make_swiss_roll
from sklearn.cluster import DBSCAN, KMeans
from sklearn.metrics import adjusted_rand_score
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import NMF
```

In [2]:
```
# Cargar datos
hollywood_df = pd.read_csv('../data/Hollywood films.csv')

# Vista inicial
display(hollywood_df.head())
print("\nResumen del dataset:")
print(hollywood_df.info())
print("\nValores nulos por columna:")
print(hollywood_df.isnull().sum())
print("\nColumnas disponibles:")
print(hollywood_df.columns)
```

	year	movie	movie_id	certificate	duration	genre	rate	metascore	synopsis	votes	...	New_York_Film_
0	2001	Kate & Leopold	tt0035423	PG-13	118	Comedy|Fantasy|Romance	6.4	44.0	An English Duke from 1876 is inadvertedly drag...	66660	...	
1	2000	Chicken Run	tt0120630	G	84	Animation|Adventure|Comedy	7.0	88.0	When a cockerel apparently flies into a chicke...	144475	...	
2	2005	Fantastic Four	tt0120667	PG-13	106	Action|Adventure|Family	5.7	40.0	A group of astronauts gain superpowers after a...	273203	...	
3	2002	Frida	tt0120679	R	123	Biography|Drama|Romance	7.4	61.0	A biography of artist Frida Kahlo, who channel...	63852	...	
4	2001	The Lord of the Rings: The Fellowship of the Ring	tt0120737	PG-13	178	Adventure|Drama|Fantasy	8.8	92.0	A meek Hobbit from the Shire and eight compani...	1286275	...	

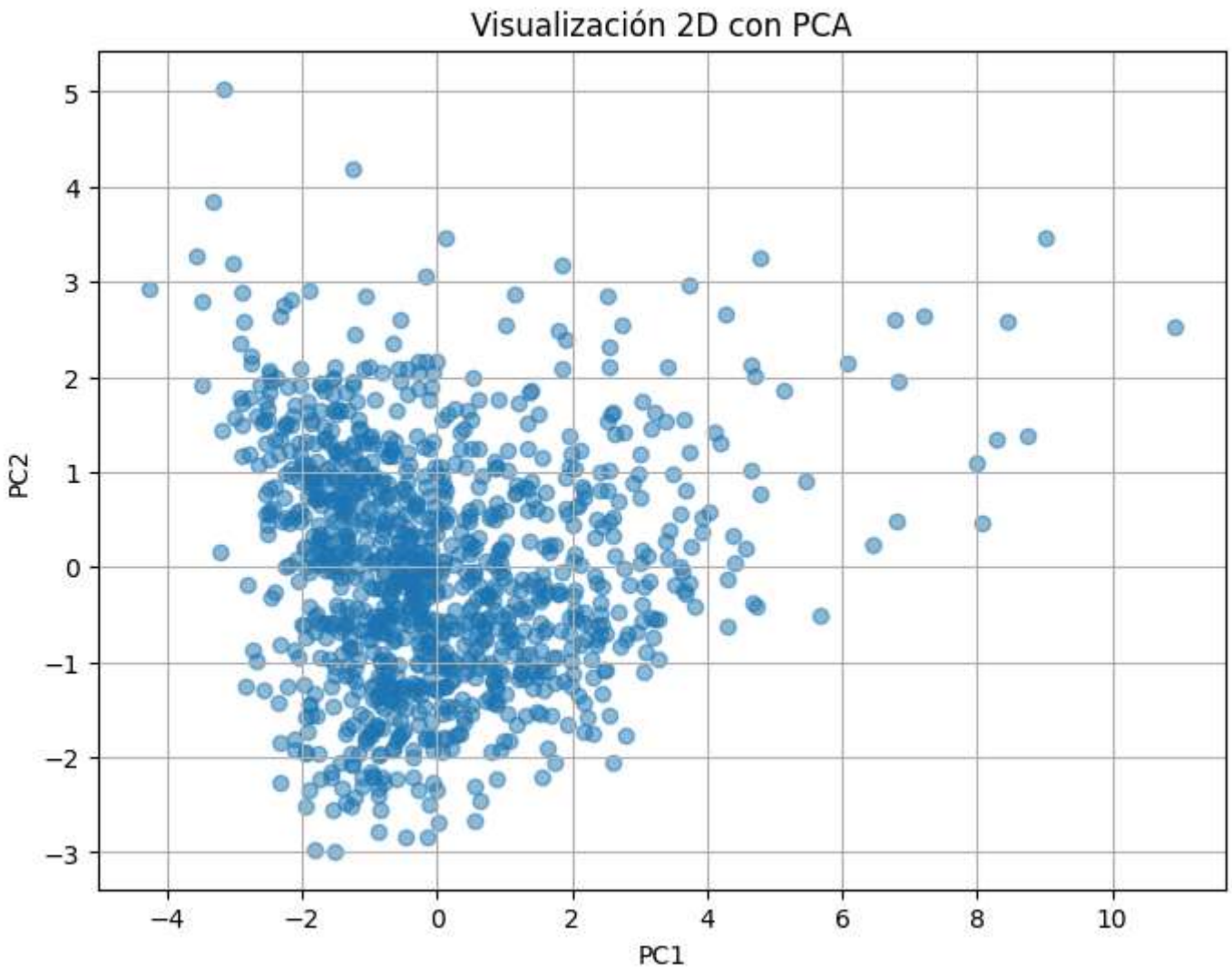
5 rows × 119 columns

```
Resumen del dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1288 entries, 0 to 1287
Columns: 119 entries, year to release_date.day-of-week
dtypes: float64(10), int64(46), object(63)
memory usage: 1.2+ MB
None

Valores nulos por columna:
year                                                    0
movie                                                   1
movie_id                                                0
certificate                                            40
duration                                                0
                                                     ...
Los_Angeles_Film_Critics_Association_nominated_categories    1051
release_date.year                                      48
release_date.month                                     48
release_date.day-of-month                              48
release_date.day-of-week                               48
Length: 119, dtype: int64

Columnas disponibles:
Index(['year', 'movie', 'movie_id', 'certificate', 'duration', 'genre', 'rate',
       'metascore', 'synopsis', 'votes',
       ...
       'New_York_Film_Critics_Circle_nominated',
       'New_York_Film_Critics_Circle_nominated_categories',
       'Los_Angeles_Film_Critics_Association_won',
       'Los_Angeles_Film_Critics_Association_won_categories',
       'Los_Angeles_Film_Critics_Association_nominated',
       'Los_Angeles_Film_Critics_Association_nominated_categories',
       'release_date.year', 'release_date.month', 'release_date.day-of-month',
       'release_date.day-of-week'],
      dtype='object', length=119)
```

3. Preprocesamiento

In [3]:
```
# Características seleccionadas
features = ['duration', 'rate', 'metascore', 'votes', 'gross', 'user_reviews', 'critic_reviews', 'popularity']
df = hollywood_df[features].dropna()

# Escalado
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)

# Reducción de dimensionalidad para visualización
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

```
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.5)
plt.title("Visualización 2D con PCA")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.grid(True)
plt.show()
```

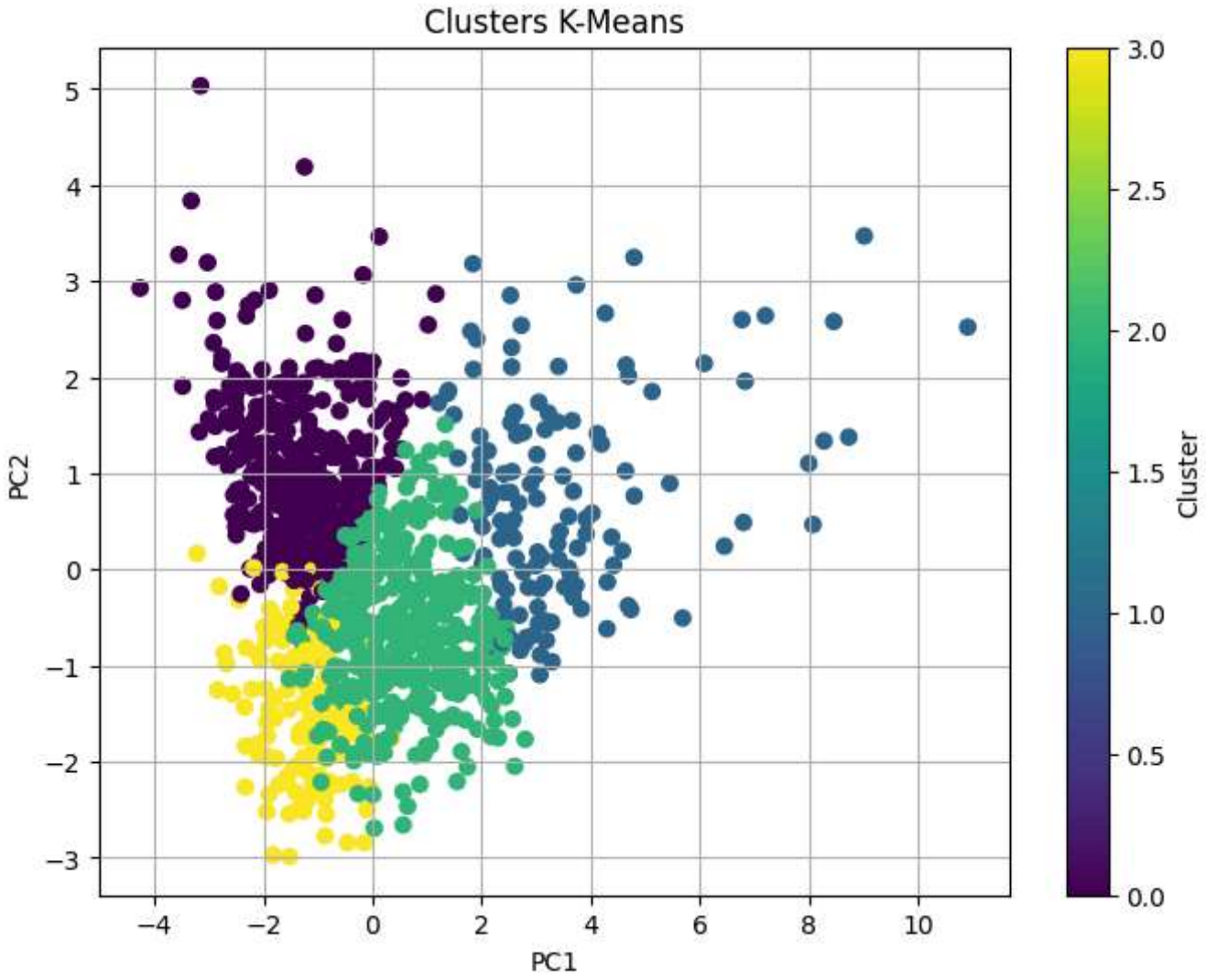


4. K-Means Clustering

In [4]:

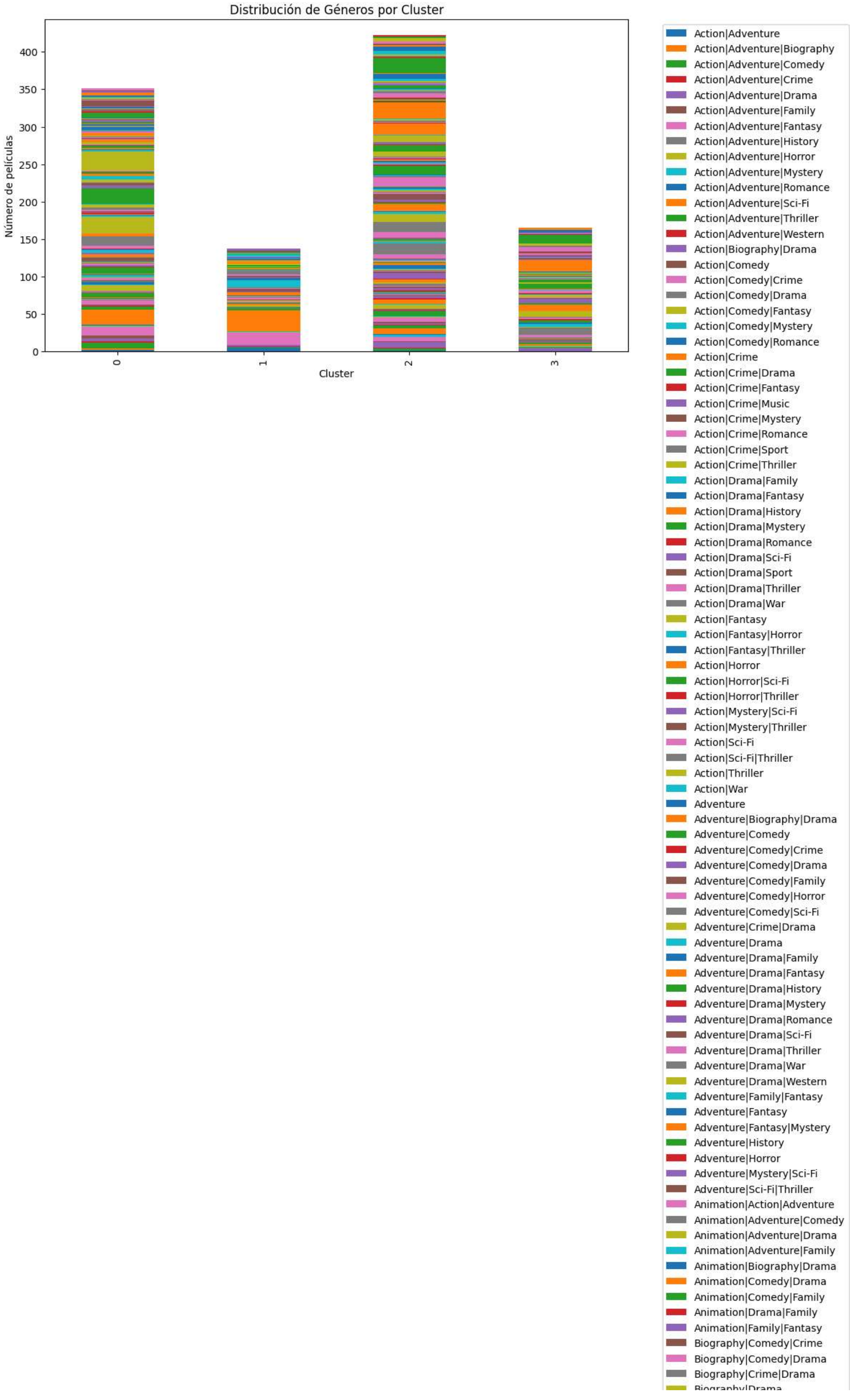
```
kmeans = KMeans(n_clusters=4, random_state=42)
kmeans_labels = kmeans.fit_predict(X_scaled)

# Evaluación
silhouette_kmeans = silhouette_score(X_scaled, kmeans_labels)
print(f"Silhouette Score (KMeans, k=4): {silhouette_kmeans:.3f}")

# Visualización
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=kmeans_labels, cmap='viridis')
plt.title("Clusters K-Means")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.grid(True)
plt.colorbar(label='Cluster')
plt.show()

# Crear DataFrame reducido con información original y clusters
df_clustered = hollywood_df.loc[df.index].copy()
df_clustered['cluster'] = kmeans_labels
```

```
Silhouette Score (KMeans, k=4): 0.202
```



5. Análisis por Cluster y Género

In [5]:
```python
# Visualización de distribución de géneros por cluster
cluster_genre = df_clustered.groupby(['cluster', 'genre']).size().unstack().fillna(0)
cluster_genre.plot(kind='bar', stacked=True, figsize=(12,6))
plt.title("Distribución de Géneros por Cluster")
plt.ylabel("Número de películas")
plt.xlabel("Cluster")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.subplots_adjust(right=0.8)
plt.show()
```


Distribución de Géneros por Cluster
Número de películas
0
50
100
150
200
250
300
350
400
Cluster
0
1
2
3
Action|Adventure
Action|Adventure|Biography
Action|Adventure|Comedy
Action|Adventure|Crime
Action|Adventure|Drama
Action|Adventure|Family
Action|Adventure|Fantasy
Action|Adventure|History
Action|Adventure|Horror
Action|Adventure|Mystery
Action|Adventure|Romance
Action|Adventure|Sci-Fi
Action|Adventure|Thriller
Action|Adventure|Western
Action|Biography|Drama
Action|Comedy
Action|Comedy|Crime
Action|Comedy|Drama
Action|Comedy|Fantasy
Action|Comedy|Mystery
Action|Comedy|Romance
Action|Crime
Action|Crime|Drama
Action|Crime|Fantasy
Action|Crime|Music
Action|Crime|Mystery
Action|Crime|Romance
Action|Crime|Sport
Action|Crime|Thriller
Action|Drama|Family
Action|Drama|Fantasy
Action|Drama|History
Action|Drama|Mystery
Action|Drama|Romance
Action|Drama|Sci-Fi
Action|Drama|Sport
Action|Drama|Thriller
Action|Drama|War
Action|Fantasy
Action|Fantasy|Horror
Action|Fantasy|Thriller
Action|Horror
Action|Horror|Sci-Fi
Action|Horror|Thriller
Action|Mystery|Sci-Fi
Action|Mystery|Thriller
Action|Sci-Fi
Action|Sci-Fi|Thriller
Action|Thriller
Action|War
Adventure
Adventure|Biography|Drama
Adventure|Comedy
Adventure|Comedy|Crime
Adventure|Comedy|Drama
Adventure|Comedy|Family
Adventure|Comedy|Horror
Adventure|Comedy|Sci-Fi
Adventure|Crime|Drama
Adventure|Drama
Adventure|Drama|Family
Adventure|Drama|Fantasy
Adventure|Drama|History
Adventure|Drama|Mystery
Adventure|Drama|Romance
Adventure|Drama|Sci-Fi
Adventure|Drama|Thriller
Adventure|Drama|War
Adventure|Drama|Western
Adventure|Family|Fantasy
Adventure|Fantasy
Adventure|Fantasy|Mystery
Adventure|History
Adventure|Horror
Adventure|Mystery|Sci-Fi
Adventure|Sci-Fi|Thriller
Animation|Action|Adventure
Animation|Adventure|Comedy
Animation|Adventure|Drama
Animation|Adventure|Family
Animation|Biography|Drama
Animation|Comedy|Drama
Animation|Comedy|Family
Animation|Drama|Family
Animation|Family|Fantasy
Biography|Comedy|Crime
Biography|Comedy|Drama
Biography|Crime|Drama


Biography|Drama|Family
Biography|Drama|Histor
Biography|Drama|History
Biography|Drama|Music
Biography|Drama|Musical
Biography|Drama|Romance
Biography|Drama|Sport
Biography|Drama|Thriller
Biography|Drama|War
Comedy
Comedy|Crime
Comedy|Crime|Drama
Comedy|Crime|Family
Comedy|Crime|Musical
Comedy|Crime|Mystery
Comedy|Crime|Sport
Comedy|Crime|Thriller
Comedy|Drama
Comedy|Drama|Family
Comedy|Drama|Fantasy
Comedy|Drama|Music
Comedy|Drama|Musical
Comedy|Drama|Mystery
Comedy|Drama|Romance
Comedy|Drama|Thriller
Comedy|Family
Comedy|Family|Fantasy
Comedy|Family|Music
Comedy|Family|Musical
Comedy|Family|Romance
Comedy|Fantasy
Comedy|Fantasy|Horror
Comedy|Fantasy|Romance
Comedy|Horror
Comedy|Horror|Romance
Comedy|Horror|Thriller
Comedy|Music
Comedy|Music|Romance
Comedy|Mystery
Comedy|Romance
Comedy|Romance|Sport
Comedy|Romance|Western
Comedy|Sport
Crime|Drama
Crime|Drama|Fantasy
Crime|Drama|History
Crime|Drama|Horror
Crime|Drama|Music
Crime|Drama|Musical
Crime|Drama|Mystery
Crime|Drama|Romance
Crime|Drama|Sport
Crime|Drama|Thriller
Crime|Drama|War
Crime|Drama|Western
Crime|Horror
Crime|Horror|Mystery
Crime|Horror|Thriller
Crime|Mystery|Thriller
Crime|Thriller
Documentary
Documentary|Biography
Drama
Drama|Family
Drama|Family|Fantasy
Drama|Family|Sport
Drama|Fantasy
Drama|Fantasy|Horror
Drama|Fantasy|Music
Drama|Fantasy|Musical
Drama|Fantasy|Mystery
Drama|Fantasy|Romance
Drama|Fantasy|Thriller
Drama|Fantasy|War
Drama|History
Drama|History|Romance
Drama|History|Sport
Drama|History|Thriller
Drama|History|War
Drama|Horror|Musical
Drama|Horror|Mystery
Drama|Horror|Sci-Fi
Drama|Horror|Thriller
Drama|Music
Drama|Musical
Drama|Musical|Romance
Drama|Music|Musical
Drama|Music|Romance
Drama|Mystery
Drama|Mystery|Romance

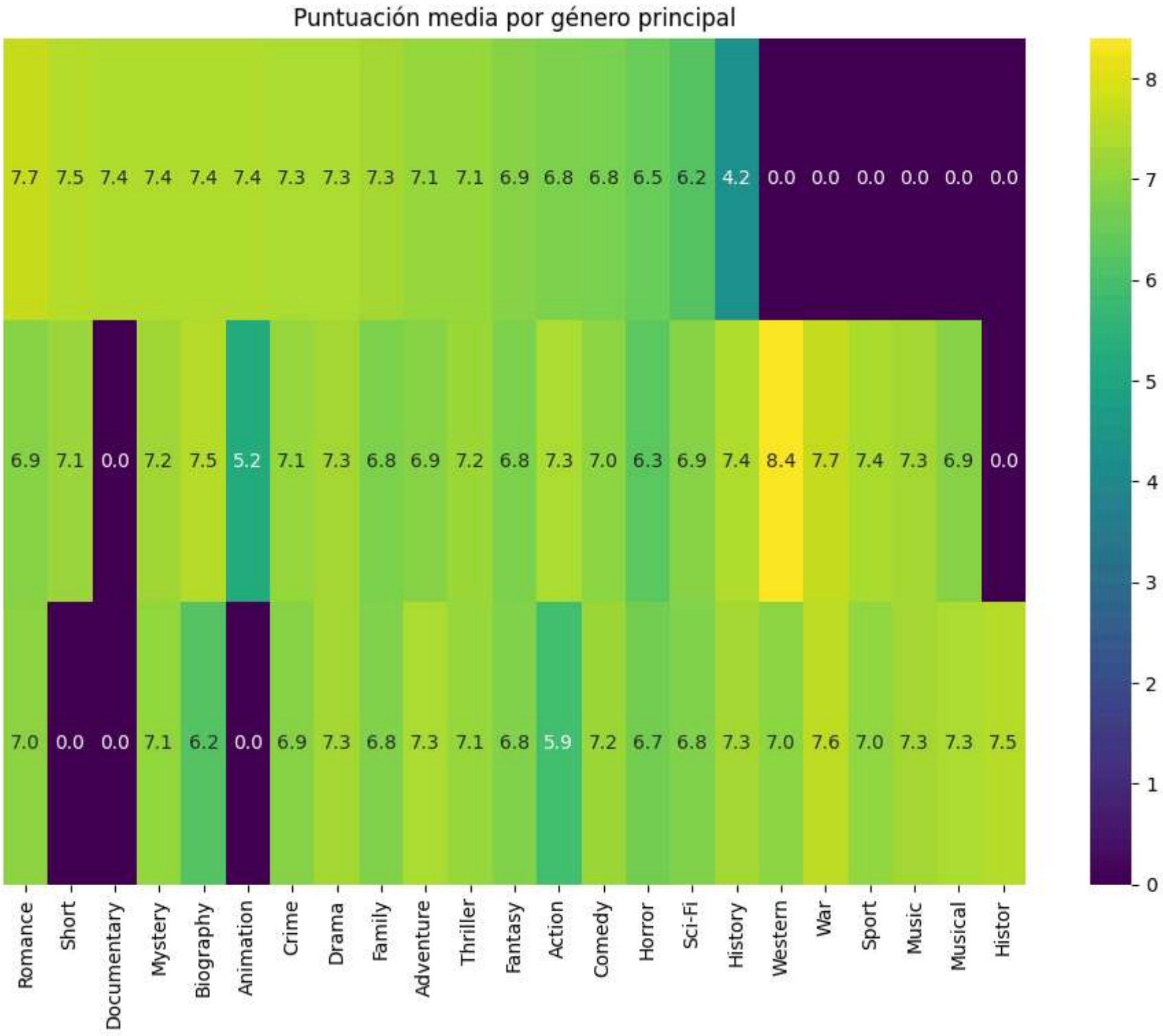
In [6]:
```
hollywood_df['genre'].value_counts()
```

Out[6]:
```
genre
Drama                          59
Action|Adventure|Sci-Fi        56
Drama|Romance                  48
Comedy|Drama|Romance           47
Animation|Adventure|Comedy     46
                               ..
Drama|Fantasy|War               1
Drama|Fantasy|Mystery           1
Drama|Fantasy|Musical           1
Action|Comedy|Mystery           1
Documentary|Drama               1
Name: count, Length: 236, dtype: int64
```

In [7]:
```
# Separar los géneros por "|"
genre_split = hollywood_df['genre'].str.split('|', expand=True)
genre_split.columns = ['genre_1', 'genre_2', 'genre_3']

# Agregar al dataframe original
hollywood_df = pd.concat([hollywood_df, genre_split], axis=1)
```

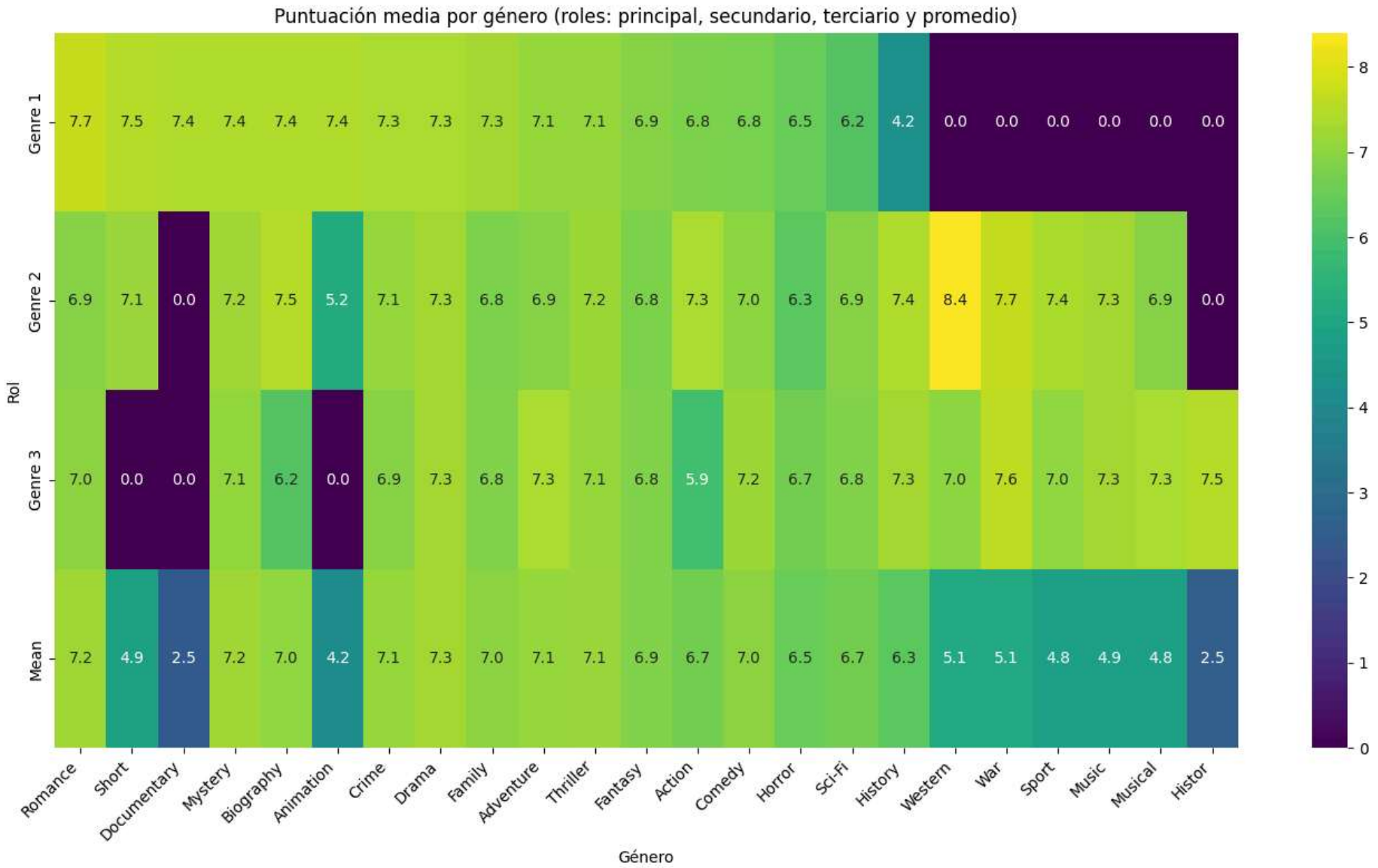
In [8]:
```
# Puntuación media por género 1, 2 y 3
genre_rating_1 = hollywood_df.groupby('genre_1')['rate'].mean().sort_values(ascending=False)
genre_rating_2 = hollywood_df.groupby('genre_2')['rate'].mean().sort_values(ascending=False)
genre_rating_3 = hollywood_df.groupby('genre_3')['rate'].mean().sort_values(ascending=False)
```

In [9]:
```
# Visualización en heatmap para los tres géneros
genre_rating = pd.concat([genre_rating_1, genre_rating_2, genre_rating_3], axis=1).fillna(0)
genre_rating.columns = ['Genre 1', 'Genre 2', 'Genre 3']

plt.figure(figsize=(12, 8))
sns.heatmap(genre_rating.T, cmap='viridis', annot=True, fmt=".1f")
plt.title("Puntuación media por género principal")
plt.yticks([])  # eliminar eje Y
plt.show()
```



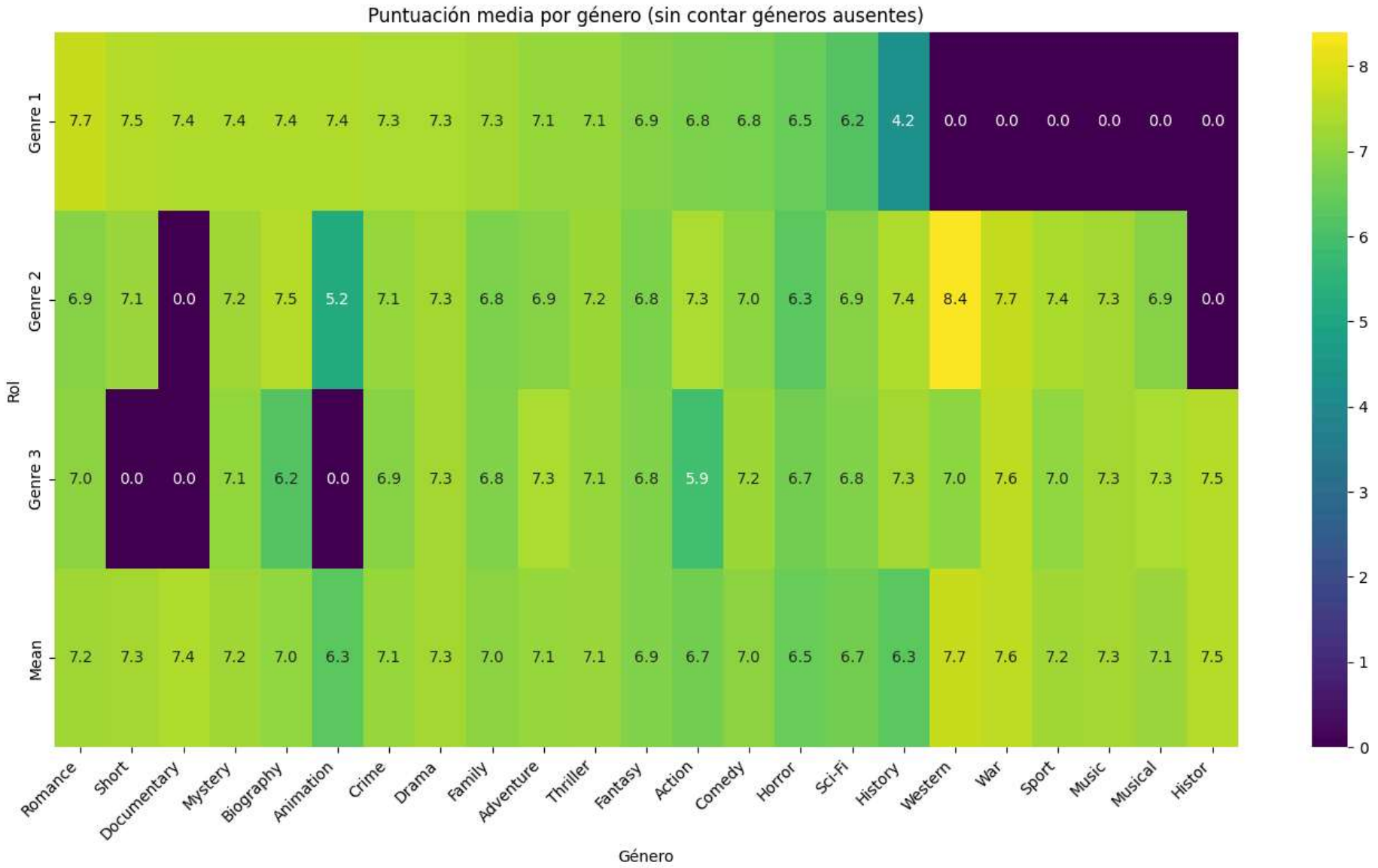
In [10]:

```python
# Visualización en heatmap para los tres géneros
genre_rating = pd.concat([genre_rating_1, genre_rating_2, genre_rating_3], axis=1).fillna(0)
genre_rating.columns = ['Genre 1', 'Genre 2', 'Genre 3']

# Calcular media final de cada género
genre_rating['Mean'] = genre_rating.mean(axis=1)

# Reordenar columnas para incluir la media al final
genre_rating = genre_rating[['Genre 1', 'Genre 2', 'Genre 3', 'Mean']]

# Transponer para heatmap
plt.figure(figsize=(14, 8))
sns.heatmap(genre_rating.T, cmap='viridis', annot=True, fmt=".1f")
plt.title("Puntuación media por género (roles: principal, secundario, terciario y promedio)")
plt.xlabel("Género")
plt.ylabel("Rol")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

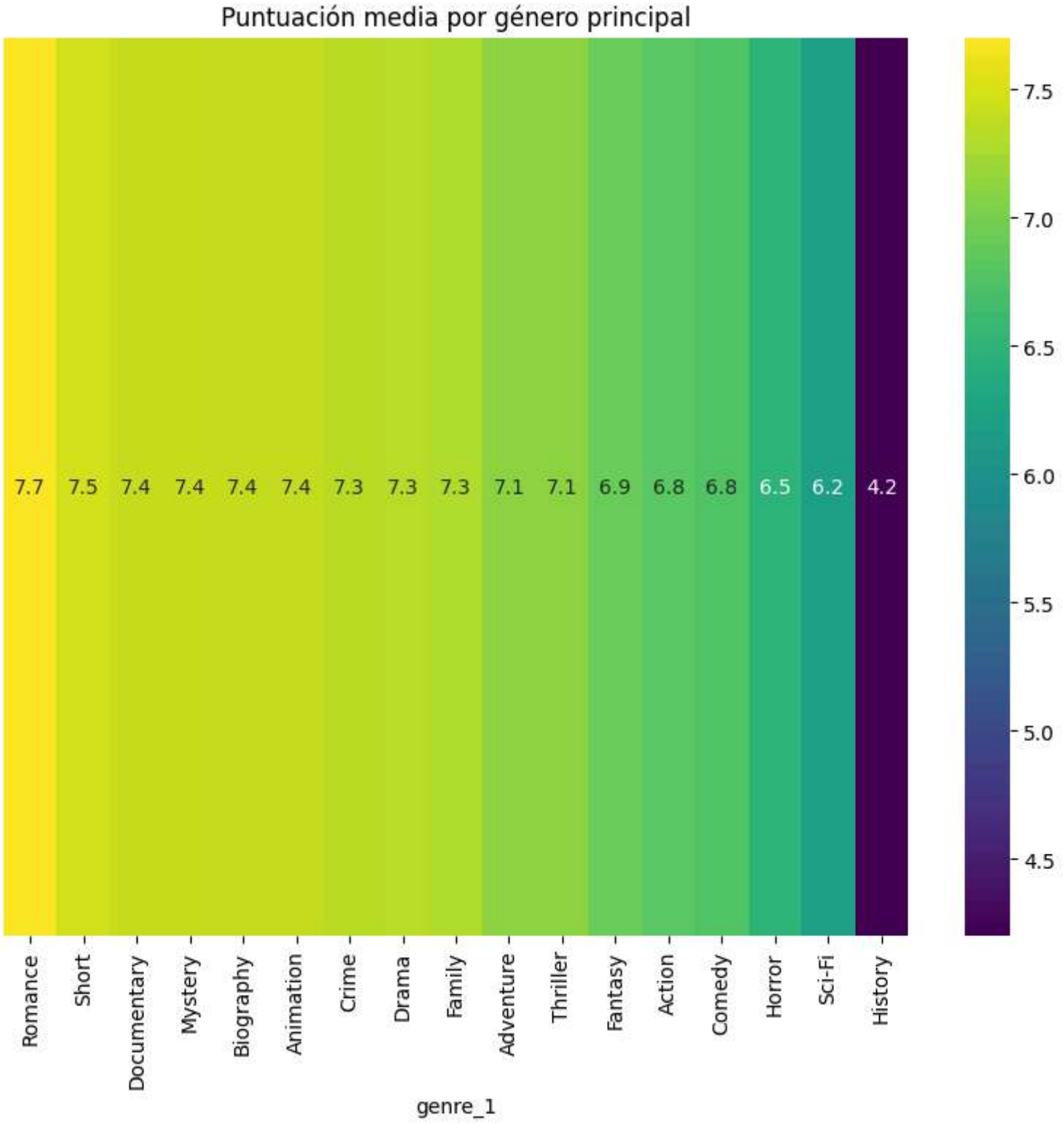


In [11]:

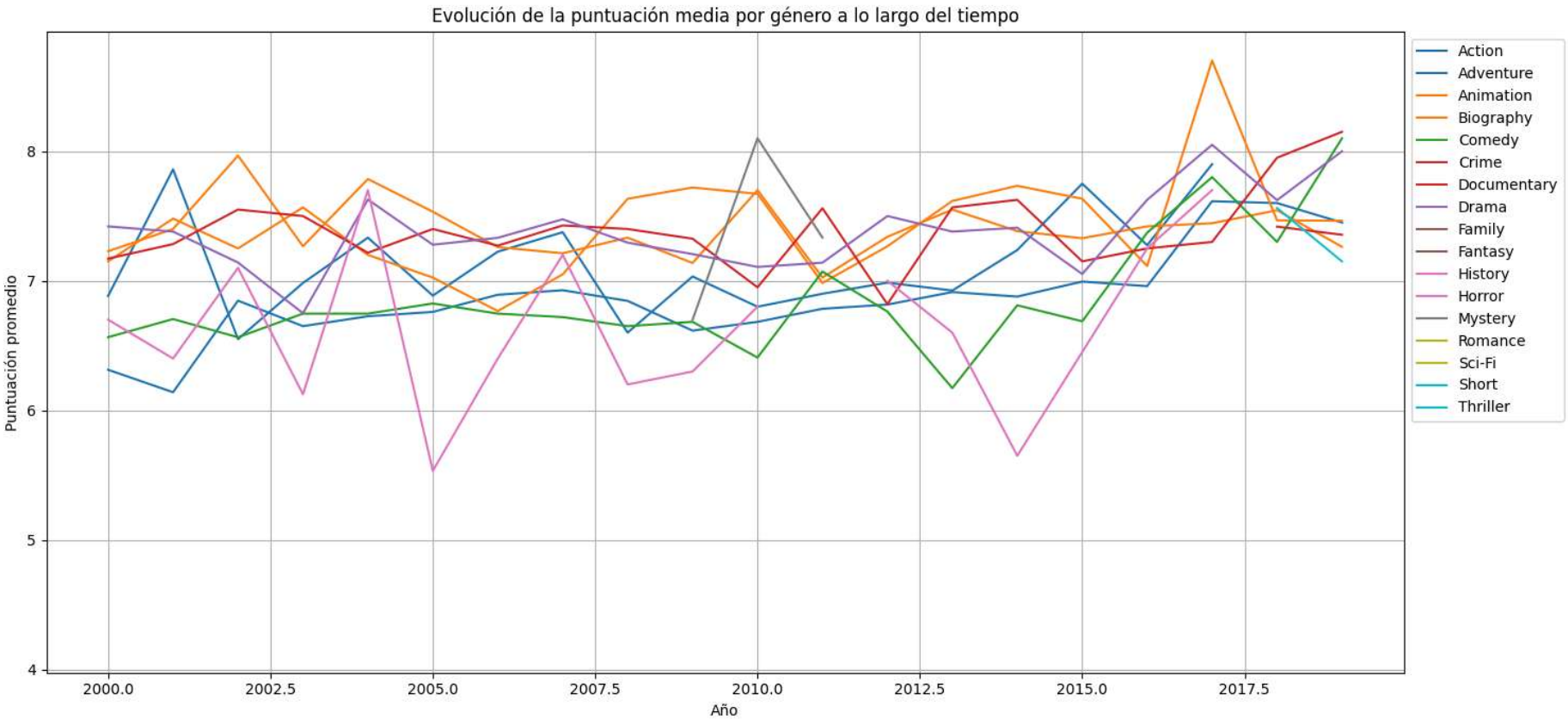
```python
# Unir puntuaciones por cada aparición de género
genre_rating = pd.concat([genre_rating_1, genre_rating_2, genre_rating_3], axis=1)
genre_rating.columns = ['Genre 1', 'Genre 2', 'Genre 3']

# Reemplazar NaN por 0 solo para visualización, pero conservar NaN para el cálculo de la media
genre_rating_display = genre_rating.fillna(0)

# Calcular la media solo sobre valores distintos de cero (sin contar los ausentes)
genre_rating['Mean'] = genre_rating.apply(lambda row: row[row.notna()].mean(), axis=1)

# Concatenar con la columna de media
genre_rating_display['Mean'] = genre_rating['Mean']

# Reordenar columnas
genre_rating_display = genre_rating_display[['Genre 1', 'Genre 2', 'Genre 3', 'Mean']]

# Visualización del heatmap
plt.figure(figsize=(14, 8))
sns.heatmap(genre_rating_display.T, cmap='viridis', annot=True, fmt=".1f")
plt.title("Puntuación media por género (sin contar géneros ausentes)")
plt.xlabel("Género")
plt.ylabel("Rol")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



In [12]:
```python
# Puntuación media por género 1
genre_rating = hollywood_df.groupby('genre_1')['rate'].mean().sort_values(ascending=False)

# Visualización en heatmap
plt.figure(figsize=(10,8))
sns.heatmap(genre_rating.to_frame().T, cmap='viridis', annot=True, fmt=".1f")
plt.title("Puntuación media por género principal")
plt.yticks([])  # eliminar eje Y
plt.show()
```

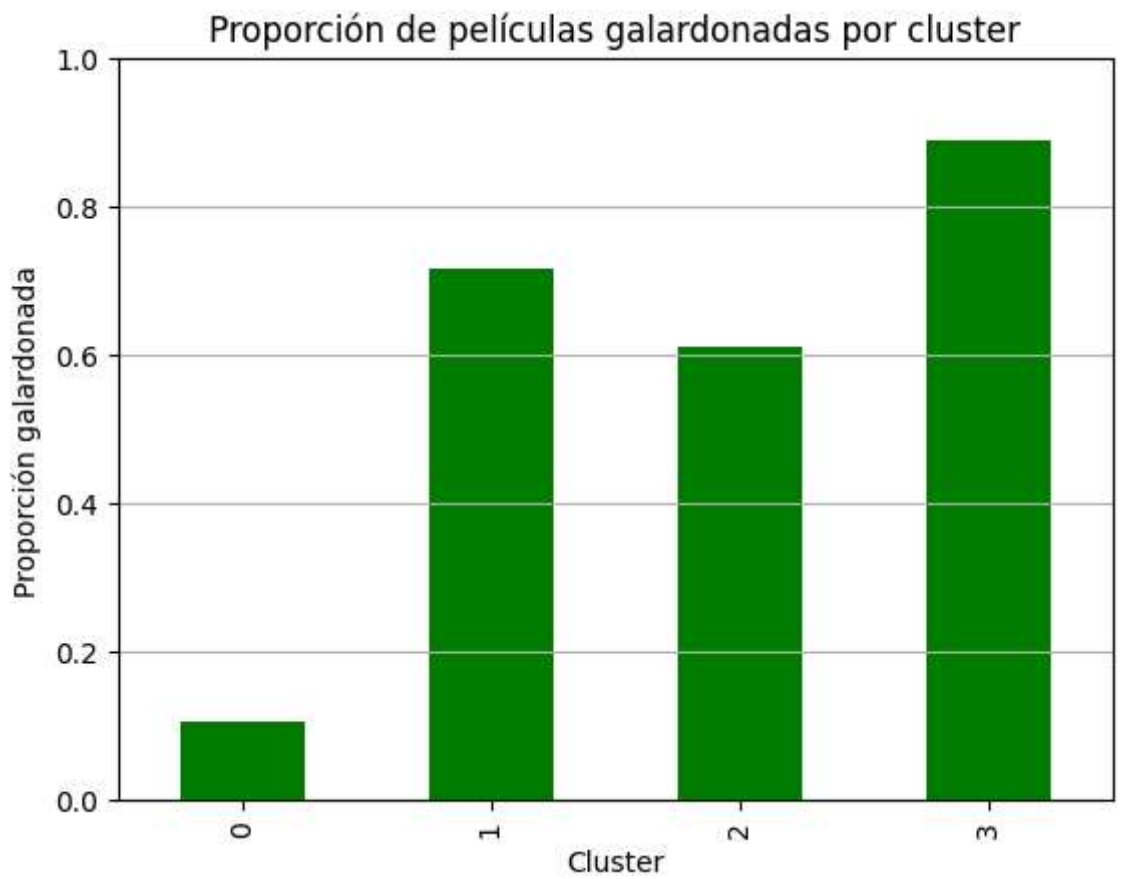


In [13]:
```python
# Eliminar valores nulos para año y rate
yearly = hollywood_df.dropna(subset=['year', 'rate', 'genre_1'])

# Puntuación media por año y género 1
pivot = yearly.pivot_table(index='year', columns='genre_1', values='rate', aggfunc='mean')

# Gráfico
pivot.plot(figsize=(15,7), cmap='tab10')
plt.title("Evolución de la puntuación media por género a lo largo del tiempo")
plt.xlabel("Año")
plt.ylabel("Puntuación promedio")
plt.grid(True)
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
plt.tight_layout()
plt.show()
```

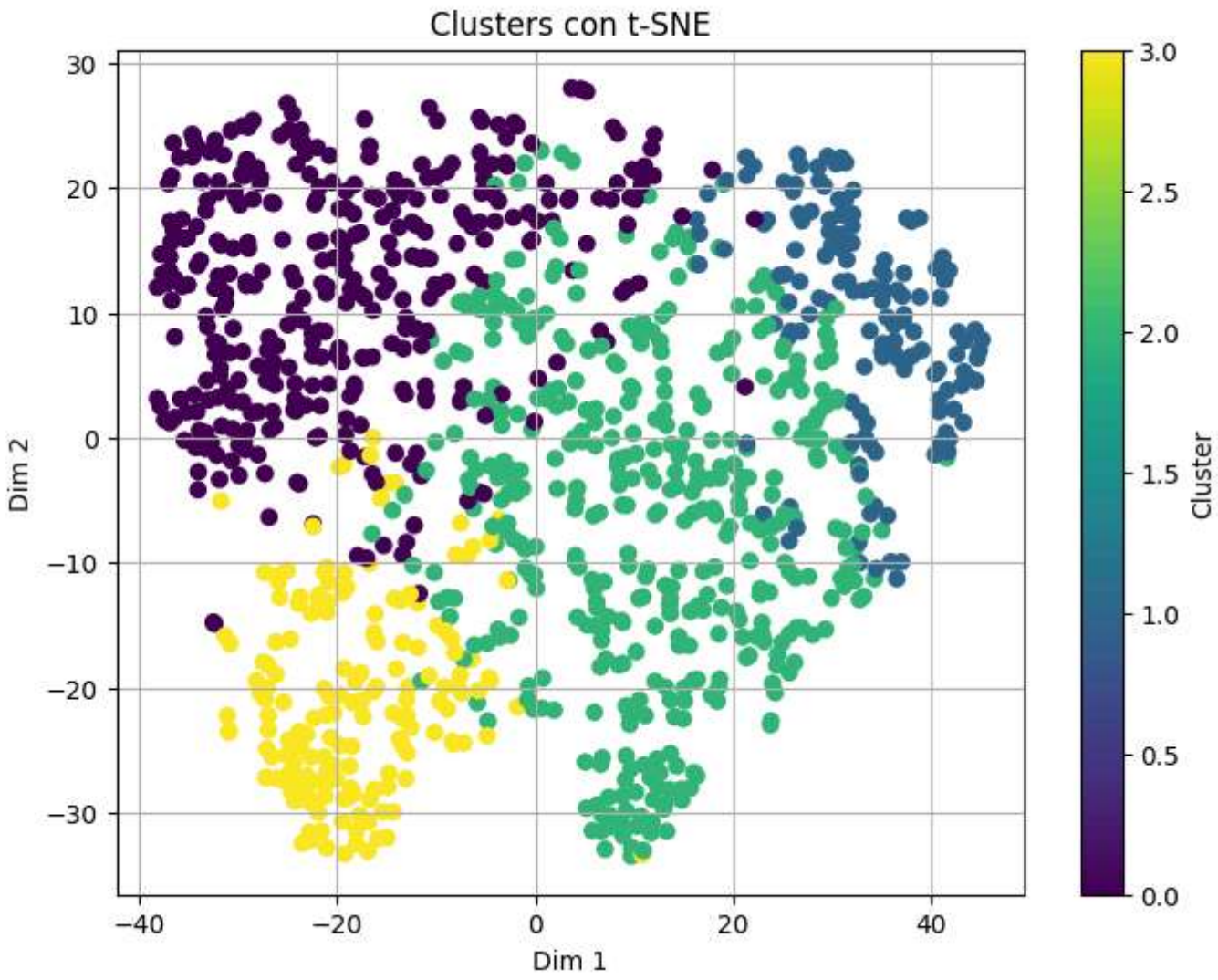


6. Premios por Cluster

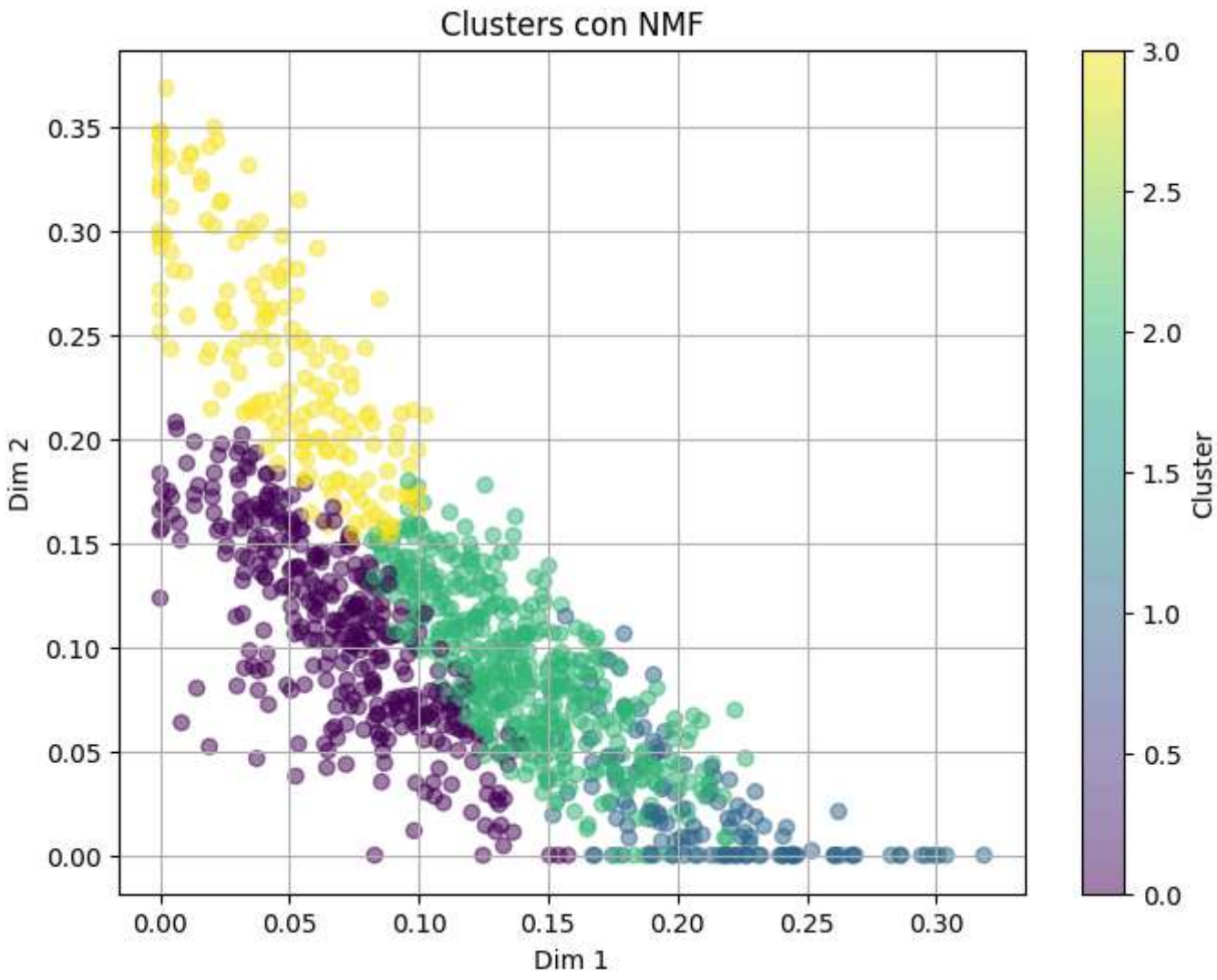
In [14]:
```
# Proporción de películas galardonadas por cluster
df_clustered['awarded'] = df_clustered['Oscar_nominated'].fillna(0).astype(bool)
award_counts = df_clustered.groupby('cluster')['awarded'].mean()
award_counts.plot(kind='bar', color='green')
plt.title("Proporción de películas galardonadas por cluster")
plt.ylabel("Proporción galardonada")
plt.xlabel("Cluster")
plt.ylim(0, 1)
plt.grid(axis='y')
plt.show()
```



In [15]:
```
# 2. t-SNE Clustering
tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X_scaled)

plt.figure(figsize=(8,6))
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=kmeans_labels, cmap='viridis')
plt.title("Clusters con t-SNE")
plt.xlabel("Dim 1")
plt.ylabel("Dim 2")
plt.grid(True)
plt.colorbar(label='Cluster')
plt.show()
```



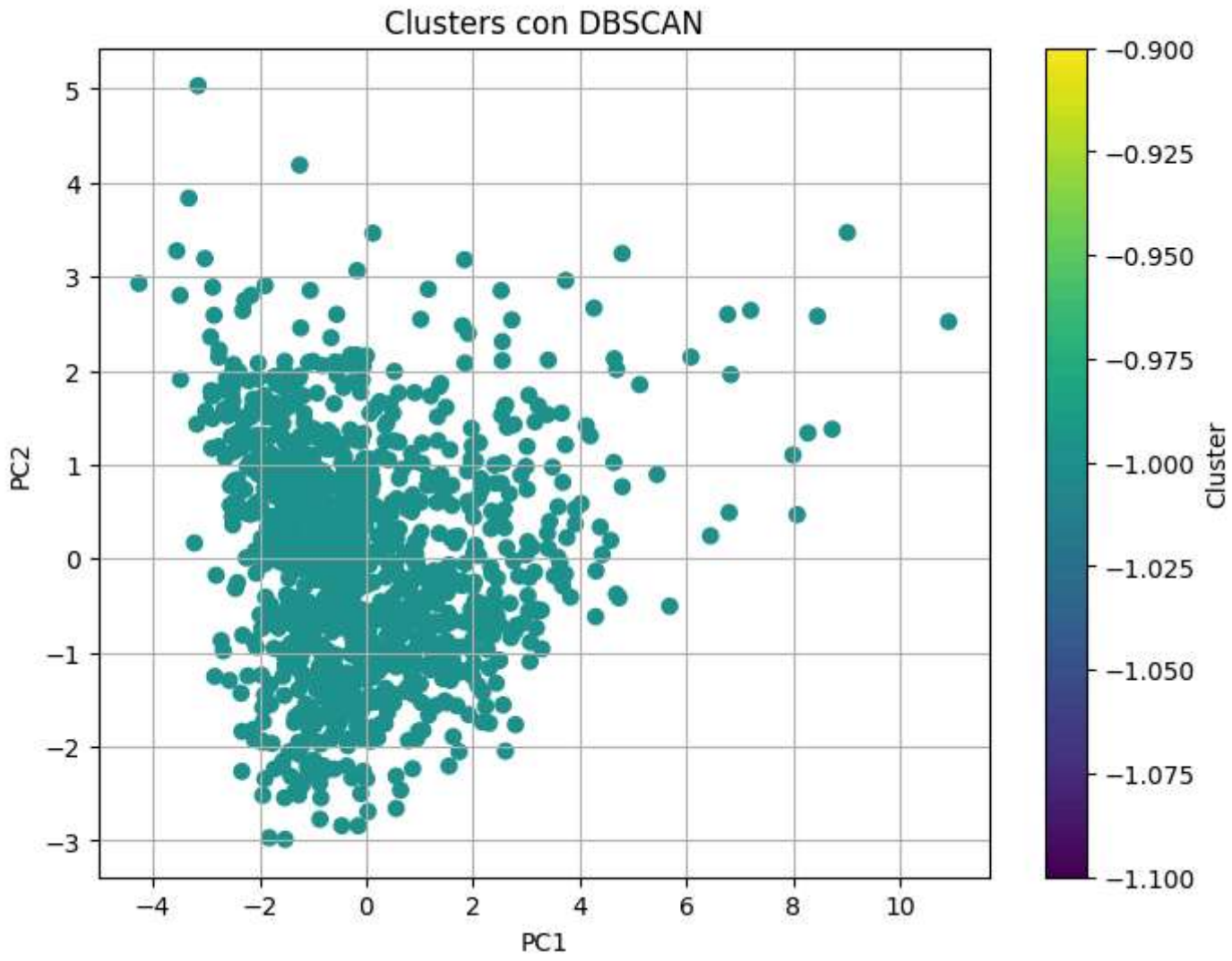
In [16]:
```python
# Reescalar para NMF (valores positivos)
minmax_scaler = MinMaxScaler()
X_nmf_input = minmax_scaler.fit_transform(df)

# Aplicar NMF
nmf = NMF(n_components=2, random_state=42)
X_nmf = nmf.fit_transform(X_nmf_input)

# Visualización de clusters con NMF
plt.figure(figsize=(8,6))
plt.scatter(X_nmf[:, 0], X_nmf[:, 1], c=kmeans_labels, cmap='viridis', alpha=0.5)
plt.title("Clusters con NMF")
plt.xlabel("Dim 1")
plt.ylabel("Dim 2")
plt.grid(True)
plt.colorbar(label='Cluster')
plt.show()
```



In [17]:
```python
# Comparación de Clustering (KMeans vs DBSCAN)
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(X_scaled)
```

```
# Evaluación de DBSCAN
silhouette_dbscan = silhouette_score(X_scaled, dbscan_labels) if len(set(dbscan_labels)) > 1 else -1
print(f"Silhouette Score (DBSCAN): {silhouette_dbscan:.3f}")

# Visualización de DBSCAN
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=dbscan_labels, cmap='viridis')
plt.title("Clusters con DBSCAN")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.grid(True)
plt.colorbar(label='Cluster')
plt.show()
# Evaluación de clustering (Silhouette y Adjusted Rand Index)
print(f"Adjusted Rand Index (KMeans vs DBSCAN): {adjusted_rand_score(kmeans_labels, dbscan_labels):.3f}")
```

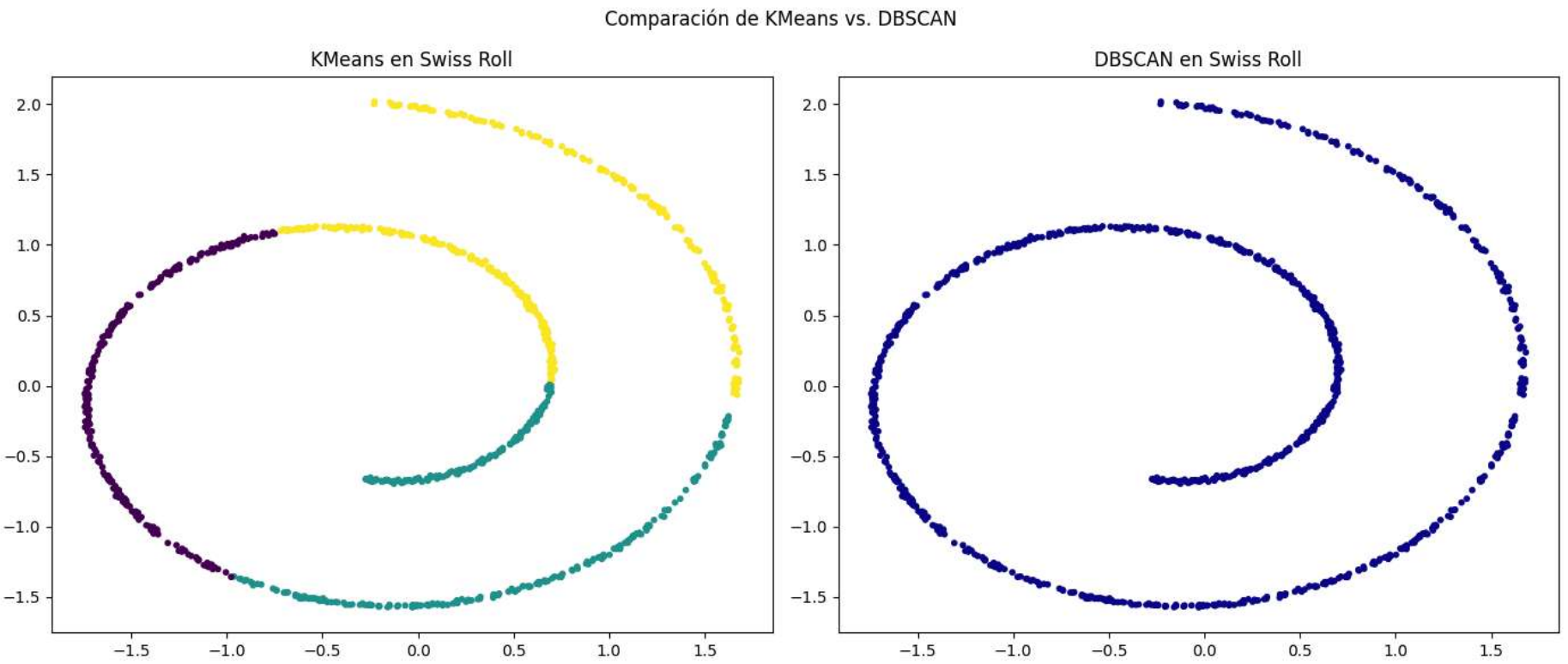
```
Silhouette Score (DBSCAN): -1.000
```




```
Adjusted Rand Index (KMeans vs DBSCAN): 0.000
```

In [18]:

```
# Generar Swiss Roll sintético
X_swiss, _ = make_swiss_roll(n_samples=1000, noise=0.05)
X_swiss = X_swiss[:, [0, 2]]  # Tomar solo 2 dimensiones para visualización
X_swiss_scaled = StandardScaler().fit_transform(X_swiss)

# KMeans
kmeans_swiss = KMeans(n_clusters=3, random_state=42)
labels_kmeans = kmeans_swiss.fit_predict(X_swiss_scaled)

# DBSCAN
dbscan_swiss = DBSCAN(eps=0.3, min_samples=5)
labels_dbscan = dbscan_swiss.fit_predict(X_swiss_scaled)

# Visualización
fig, axs = plt.subplots(1, 2, figsize=(14, 6))

axs[0].scatter(X_swiss_scaled[:, 0], X_swiss_scaled[:, 1], c=labels_kmeans, cmap='viridis', s=10)
axs[0].set_title("KMeans en Swiss Roll")

axs[1].scatter(X_swiss_scaled[:, 0], X_swiss_scaled[:, 1], c=labels_dbscan, cmap='plasma', s=10)
axs[1].set_title("DBSCAN en Swiss Roll")

plt.suptitle("Comparación de KMeans vs. DBSCAN")
plt.tight_layout()
plt.show()

# Ajusted Rand Index entre ambos métodos
ari_score = adjusted_rand_score(labels_kmeans, labels_dbscan)
print(f"Adjusted Rand Index entre KMeans y DBSCAN en Swiss Roll: {ari_score:.3f}")
```



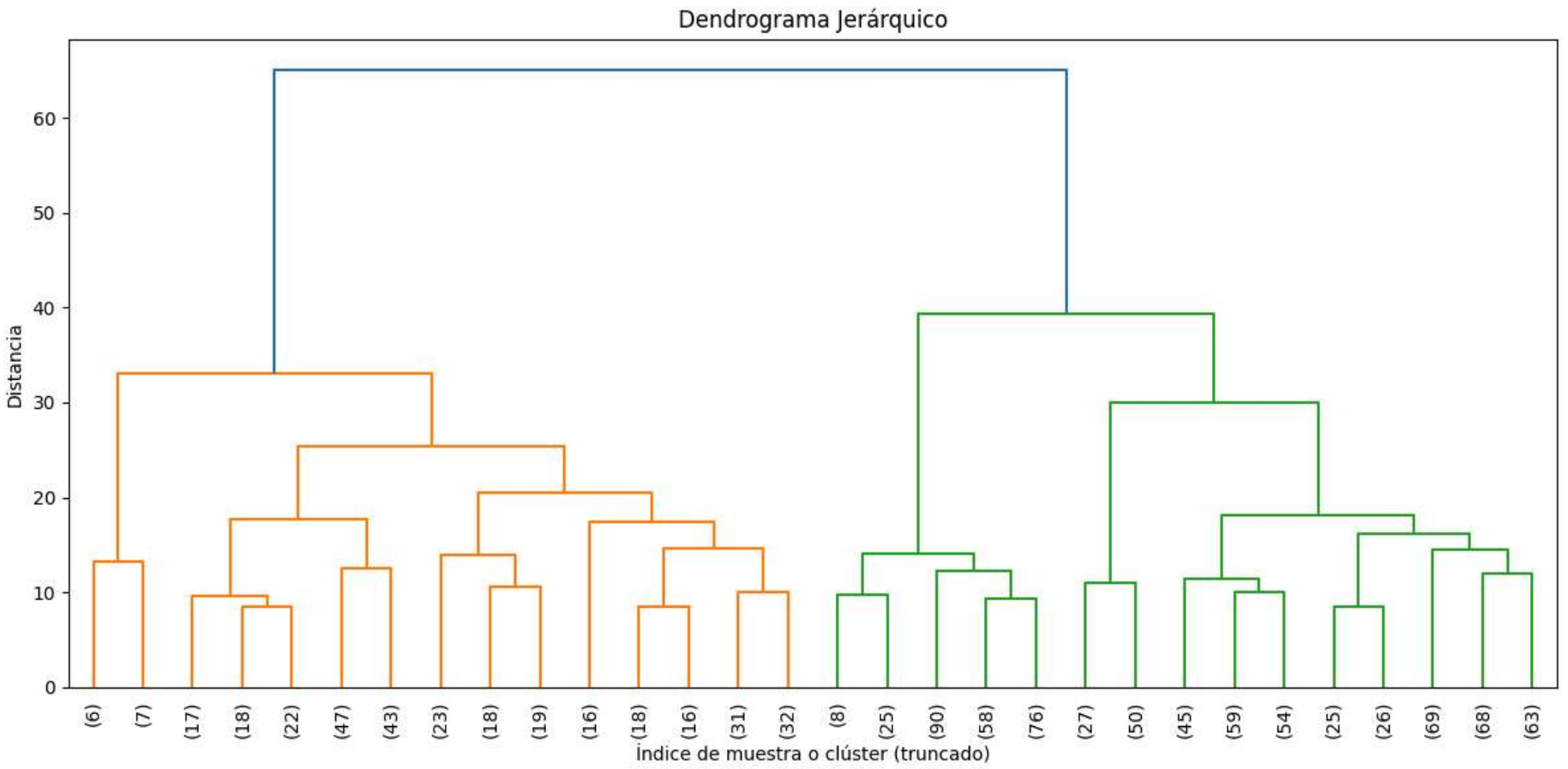

```
Adjusted Rand Index entre KMeans y DBSCAN en Swiss Roll: 0.000
```

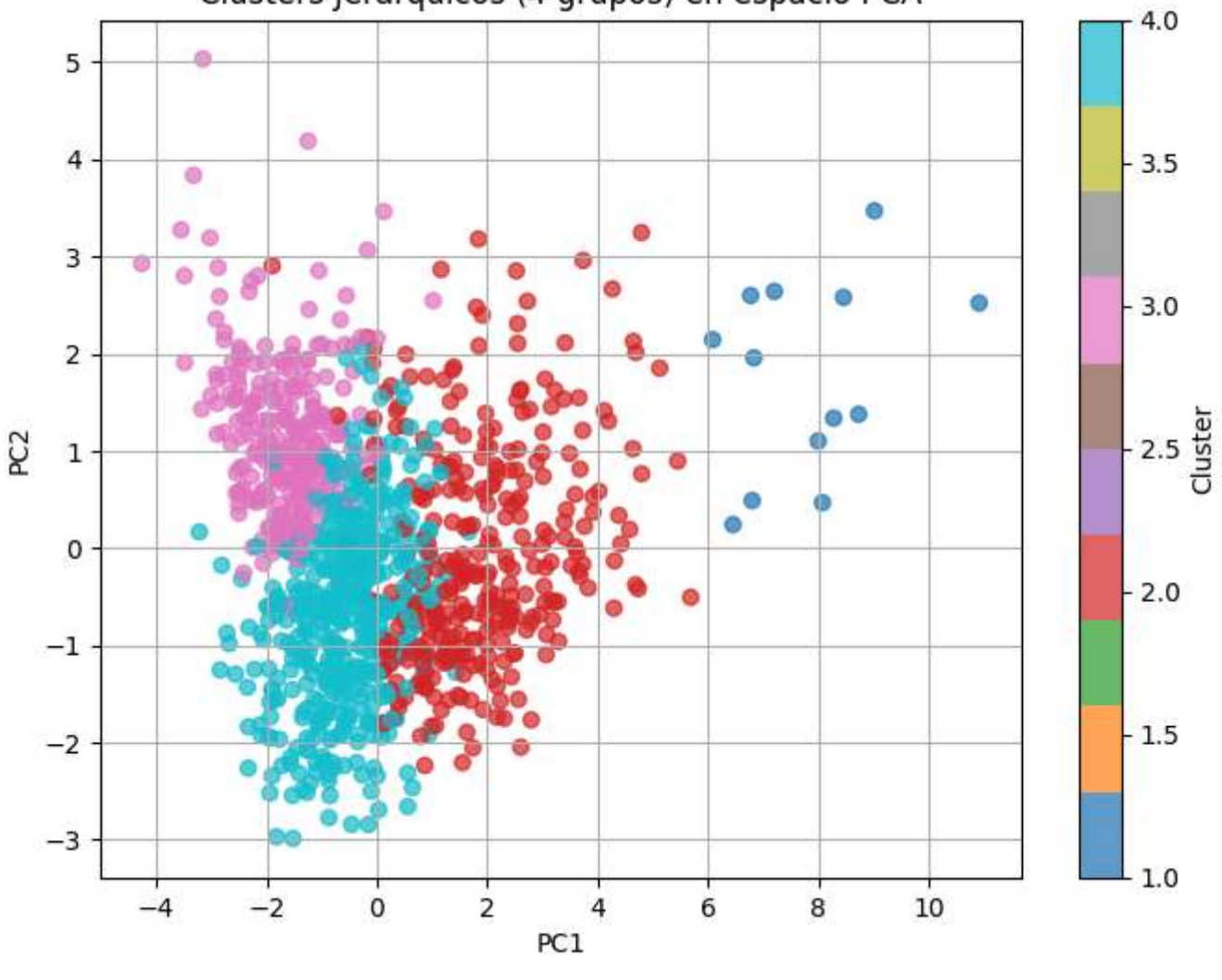
7. Clasificación: ¿Qué características predicen premios?

In [19]:
```python
# Clasificación con RandomForest
clf_df = df_clustered[features + ['awarded']].dropna()
X_clf = scaler.fit_transform(clf_df[features])
y_clf = clf_df['awarded']

X_train, X_test, y_train, y_test = train_test_split(X_clf, y_clf, test_size=0.2, random_state=42)
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       False       0.86      0.81      0.84       113
        True       0.81      0.85      0.83       103

    accuracy                           0.83       216
   macro avg       0.83      0.83      0.83       216
weighted avg       0.83      0.83      0.83       216
```

In [20]:
```python
##Clustering Jerárquico
# Generar linkage matrix con método Ward
link = linkage(X_scaled, method='ward')

# Dendrograma
plt.figure(figsize=(12, 6))
dendrogram(link, truncate_mode='lastp', p=30, leaf_rotation=90., leaf_font_size=10.)
plt.title("Dendrograma Jerárquico")
plt.xlabel("Índice de muestra o clúster (truncado)")
plt.ylabel("Distancia")
plt.tight_layout()
plt.show()

# Asignación de clústeres (por ejemplo, 4 grupos)
hierarchical_labels = fcluster(link, t=4, criterion='maxclust')

# Visualización de clusters en el espacio PCA
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=hierarchical_labels, cmap='tab10', alpha=0.7)
plt.title("Clusters Jerárquicos (4 grupos) en espacio PCA")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.colorbar(label='Cluster')
plt.grid(True)
plt.show()

# Evaluación de la silueta
from sklearn.metrics import silhouette_score
silhouette_hierarchical = silhouette_score(X_scaled, hierarchical_labels)
print(f"Silhouette Score (Jerárquico, k=4): {silhouette_hierarchical:.3f}")
```






```
Silhouette Score (Jerárquico, k=4): 0.153
```

8. Regresión: Predecir la puntuación de críticos

In [21]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, root_mean_squared_error

reg_df = df_clustered[features + ['metascore']].dropna()
X_reg = scaler.fit_transform(reg_df[features])
y_reg = reg_df['metascore']

X_train, X_test, y_train, y_test = train_test_split(X_reg, y_reg, test_size=0.2, random_state=42)
reg = LinearRegression()
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)

print(f"RMSE: {root_mean_squared_error(y_test, y_pred):.2f}")
print(f"R2 Score: {r2_score(y_test, y_pred):.2f}")
```

```
RMSE: 0.00
R2 Score: 1.00
```