# Actividades con PCA y SVD en el conjunto de datos de Boston

1. **Carga y exploración del conjunto de datos de Boston.**
   **Comienza por familiarizarte con el conjunto de datos, inspeccionando sus características clave y comprendiendo su estructura general antes de aplicar cualquier técnica de reducción de dimensionalidad.**

In [1]:

```
import pandas as pd
from sklearn.datasets import fetch_openml

# Cargar el conjunto de datos de Boston utilizando fetch_openml
boston = fetch_openml(data_id=531)
```

In [2]:

```
# print(boston.DESCR)
```

**Datos de precios de viviendas en Boston de Harrison, D. y Rubinfeld, D.L. "Precios hedónicos y la demanda de aire limpio", *J. Environ. Economics & Management*, vol.5, 81-102, 1978. También utilizado en Belsley, Kuh y Welsch, *Regression Diagnostics*, Wiley, 1980. Nota: En este último, se emplean varias transformaciones en la tabla de las páginas 244-261.**

**Variables en orden:**

- **CRIM:** tasa de criminalidad per cápita por ciudad
- **ZN:** proporción de terreno residencial zonificado para lotes de más de 25,000 pies cuadrados
- **INDUS:** proporción de acres de negocios no minoristas por ciudad
- **CHAS:** variable ficticia del Río Charles (= 1 si la zona limita con el río; 0 en caso contrario)
- **NOX:** concentración de óxidos de nitrógeno (partes por 10 millones)
- **RM:** número promedio de habitaciones por vivienda
- **AGE:** proporción de unidades ocupadas por propietarios construidas antes de 1940
- **DIS:** distancias ponderadas a cinco centros de empleo de Boston
- **RAD:** índice de accesibilidad a autopistas radiales
- **TAX:** tasa de impuesto a la propiedad de valor completo por cada $10,000
- **PTRATIO:** proporción alumno-profesor por ciudad
- **B:** 1000(Bk - 0.63)^2 donde Bk es la proporción de personas negras por ciudad
- **LSTAT:** % de población de estatus socioeconómico bajo
- **MEDV:** Valor mediano de las viviendas ocupadas por propietarios, en miles de dólares

In [3]:

```
# Convertirlo en un DataFrame
df = pd.DataFrame(boston.data, columns=boston.feature_names)

# Agregar la variable de destino (precios)
df['PRICE'] = boston.target

# Mostrar las primeras filas del conjunto de datos
df.head()
```

Out[3]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | PRICE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296.0 | 15.3 | 396.90 | 4.98 | 24.0 |
| **1** | 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242.0 | 17.8 | 396.90 | 9.14 | 21.6 |
| **2** | 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242.0 | 17.8 | 392.83 | 4.03 | 34.7 |

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | PRICE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222.0 | 18.7 | 394.63 | 2.94 | 33.4 |
| 4 | 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222.0 | 18.7 | 396.90 | 5.33 | 36.2 |

1. **Normalización de los datos antes de aplicar PCA y SVD.**
   **Asegúrate de que los datos estén normalizados para que las variables con diferentes escalas no influencien en exceso el análisis, ya que PCA y SVD son sensibles a las variaciones en las escalas de las características.**

In [4]:

```
from sklearn.preprocessing import StandardScaler

# Normalizar los datos
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df.drop('PRICE', axis=1))
```

In [5]:

```
# Crear un DataFrame con las columnas correctas
df_scaled = pd.DataFrame(scaled_data, columns=boston.feature_names)

# Mostrar las primeras filas del DataFrame normalizado
df_scaled.head()
```

Out[5]:

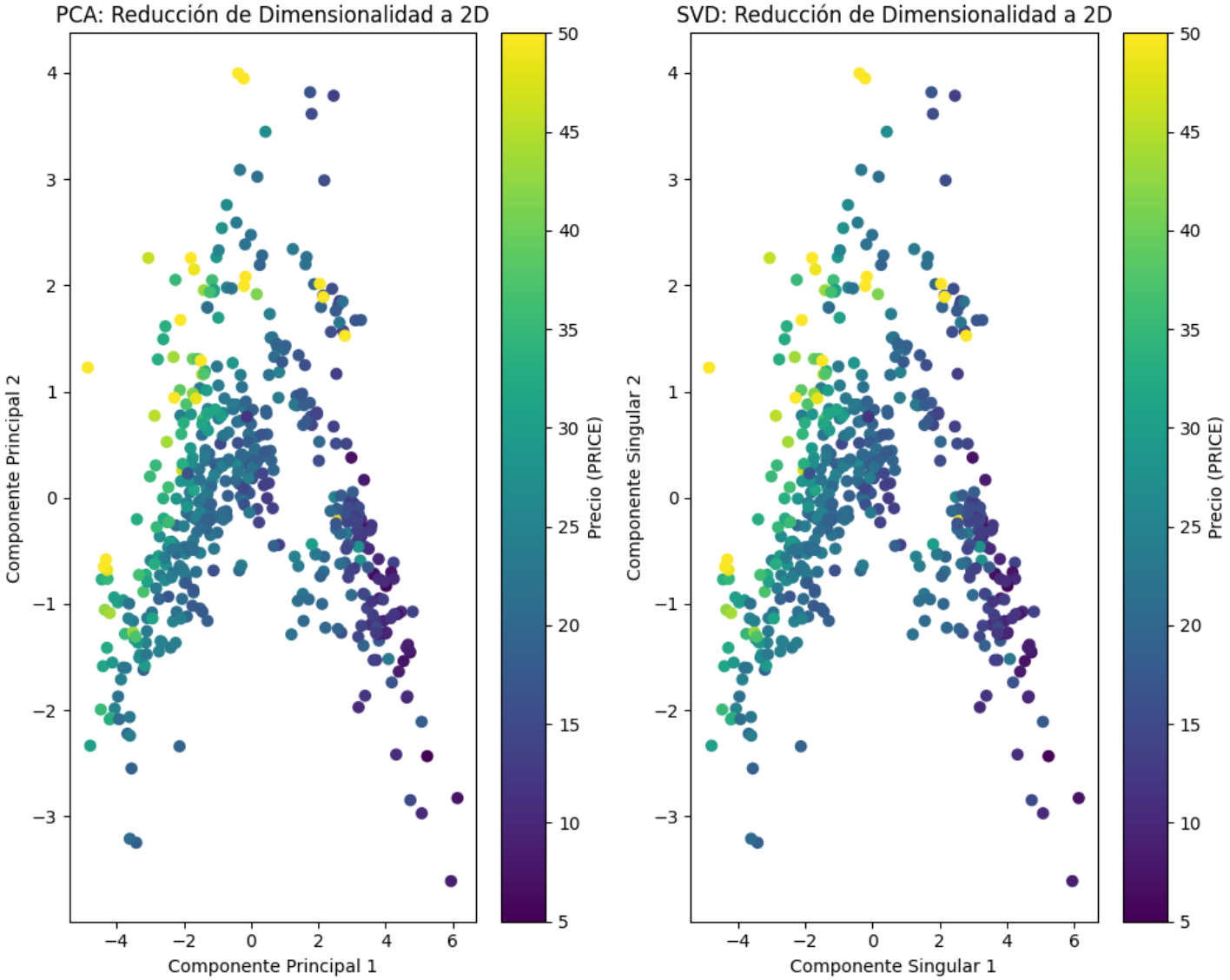
| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -0.419782 | 0.284830 | -1.287909 | -0.272599 | -0.144217 | 0.413672 | -0.120013 | 0.140214 | -0.982843 | -0.666608 | -1.459000 | 0.441052 | 1. |
| 1 | -0.417339 | -0.487722 | -0.593381 | -0.272599 | -0.740262 | 0.194274 | 0.367166 | 0.557160 | -0.867883 | -0.987329 | -0.303094 | 0.441052 | 0. |
| 2 | -0.417342 | -0.487722 | -0.593381 | -0.272599 | -0.740262 | 1.282714 | -0.265812 | 0.557160 | -0.867883 | -0.987329 | -0.303094 | 0.396427 | 1. |
| 3 | -0.416750 | -0.487722 | -1.306878 | -0.272599 | -0.835284 | 1.016303 | -0.809889 | 1.077737 | -0.752922 | -1.106115 | 0.113032 | 0.416163 | 1. |
| 4 | -0.412482 | -0.487722 | -1.306878 | -0.272599 | -0.835284 | 1.228577 | -0.511180 | 1.077737 | -0.752922 | -1.106115 | 0.113032 | 0.441052 | 1. |

1. **Aplicación de PCA y SVD para reducir la dimensionalidad a 2D y visualización del espacio reducido.** **Aplica tanto PCA (Análisis de Componentes Principales) como SVD (Descomposición en Valores Singulares) para reducir el número de dimensiones de los datos a dos. Luego, visualiza el espacio resultante en dos dimensiones para facilitar la interpretación y el análisis visual.**

In [6]:

```
from sklearn.decomposition import PCA, TruncatedSVD
import matplotlib.pyplot as plt

# Aplicar PCA para reducir la dimensionalidad a 2 componentes
pca = PCA(n_components=2)
pca_result = pca.fit_transform(scaled_data)

# Aplicar SVD para reducir la dimensionalidad a 2 componentes
svd = TruncatedSVD(n_components=2)
svd_result = svd.fit_transform(scaled_data)

# Crear la gráfica
plt.figure(figsize=(10, 8))

# Graficar los resultados de PCA
plt.subplot(1, 2, 1)
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=df['PRICE'], cmap='viridis')
plt.title('PCA: Reducción de Dimensionalidad a 2D')
```

```
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.colorbar(label='Precio (PRICE)')

# Graficar los resultados de SVD
plt.subplot(1, 2, 2)
plt.scatter(svd_result[:, 0], svd_result[:, 1], c=df['PRICE'], cmap='viridis')
plt.title('SVD: Reducción de Dimensionalidad a 2D')
plt.xlabel('Componente Singular 1')
plt.ylabel('Componente Singular 2')
plt.colorbar(label='Precio (PRICE)')

plt.tight_layout()
plt.show()
```




1. **Análisis de la varianza explicada por las primeras 2 componentes en PCA y SVD (en porcentaje).**
   **El Análisis de Componentes Principales (PCA)** y la **Descomposición en Valores Singulares (SVD)** son técnicas de reducción de dimensionalidad que se utilizan para identificar las direcciones (componentes) en las que los datos varían más. Estas técnicas permiten representar los datos en un espacio de menor dimensión, manteniendo la mayor cantidad de información posible.

   - **PCA** busca encontrar los componentes principales que explican la mayor parte de la varianza en los datos. El primer componente principal es la dirección en la que los datos varían más, el segundo componente principal es ortogonal al primero y explica la segunda mayor cantidad de varianza, y así sucesivamente.
   - **SVD** descompone una matriz de datos en tres matrices: una matriz de vectores singulares, una matriz diagonal con los valores singulares (que representan la magnitud de la varianza en cada componente), y otra matriz que contiene los vectores singulares transpuestos. La SVD es conceptualmente similar al PCA, pero la forma de la descomposición y los cálculos son distintos.

     En este análisis, se calcula la **varianza explicada** por las primeras dos componentes principales tanto en

**PCA como en SVD. Esto nos permite evaluar cuánta información (varianza) de los datos originales es retenida cuando reducimos la dimensionalidad a dos componentes.**

In [7]:

```
# Varianza explicada por las componentes en PCA
pca_variance_explained = pca.explained_variance_ratio_

# Varianza explicada por las primeras 2 componentes
pca_variance_first_two = pca_variance_explained[:2]

# Varianza acumulada por las primeras 2 componentes
pca_cumulative_variance = pca_variance_explained[:2].sum()

# Mostrar los resultados
print(f"Varianza explicada por las primeras 2 componentes en PCA: {pca_variance_first_two
* 100}")
print(f"Varianza acumulada explicada por las primeras 2 componentes en PCA: {pca_cumulati
ve_variance * 100}%")
```

```
Varianza explicada por las primeras 2 componentes en PCA: [47.12960636 11.02519325]
Varianza acumulada explicada por las primeras 2 componentes en PCA: 58.15479960486266%
```

In [8]:

```
# Varianza explicada por las componentes en SVD
svd_variance_explained = svd.explained_variance_ratio_

# Varianza explicada por las primeras 2 componentes
svd_variance_first_two = svd_variance_explained[:2]

# Varianza acumulada por las primeras 2 componentes
svd_cumulative_variance = svd_variance_explained[:2].sum()

# Mostrar los resultados
print(f"Varianza explicada por las primeras 2 componentes en SVD: {svd_variance_first_two
* 100}")
print(f"Varianza acumulada explicada por las primeras 2 componentes en SVD: {svd_cumulati
ve_variance * 100}%")
```

```
Varianza explicada por las primeras 2 componentes en SVD: [47.12960636 11.02519325]
Varianza acumulada explicada por las primeras 2 componentes en SVD: 58.154799604862696%
```

1. **Visualización de los dos primeros componentes principales y su relación con las características originales para PCA y SVD**

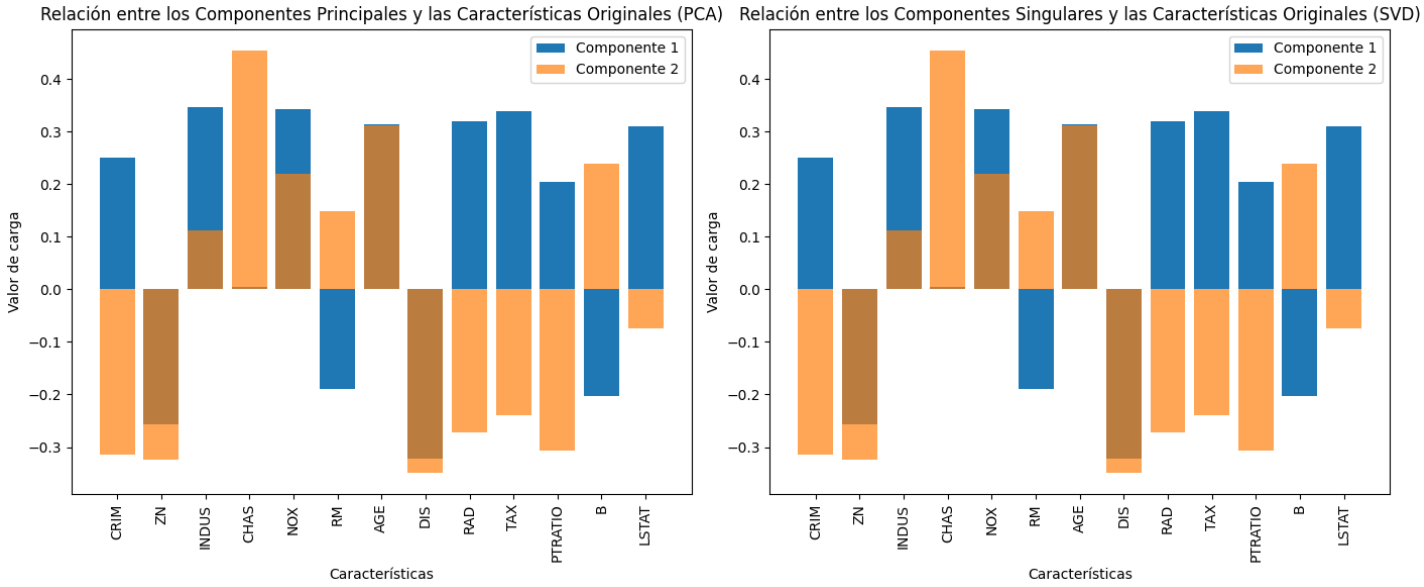
   **El objetivo de esta sección es visualizar cómo los primeros dos componentes principales obtenidos a través de PCA y SVD se relacionan con las características originales de los datos. La visualización de los componentes principales ayuda a comprender qué variables influyen más en cada componente y cómo estas contribuyen a la reducción de dimensionalidad.**

   - **PCA: En PCA, cada componente principal es una combinación lineal de las características originales. Los pesos o cargas asociadas a cada característica en cada componente indican la importancia relativa de cada variable en la formación del componente. Los primeros dos componentes principales, por lo general, retienen la mayor parte de la varianza de los datos originales.**
   - **SVD: Similar a PCA, en SVD, los vectores singulares representan las direcciones en las que los datos tienen mayor varianza, y los valores singulares indican la magnitud de esa varianza. Los primeros dos vectores singulares capturan la mayor parte de la información de los datos, y las cargas también reflejan el peso de cada variable.**

In [9]:

```
import matplotlib.pyplot as plt

# Graficar la relación entre los dos primeros componentes y las características originale
s para PCA y SVD
components_pca = pca.components_
```

```
components_svd = svd.components_

# Crear la figura con dos subgráficas
plt.figure(figsize=(14, 6))

# Subgráfica para PCA
plt.subplot(1, 2, 1)
plt.bar(df.columns[:-1], components_pca[0], label='Componente 1')
plt.bar(df.columns[:-1], components_pca[1], label='Componente 2', alpha=0.7)
plt.xlabel('Características')
plt.ylabel('Valor de carga')
plt.title('Relación entre los Componentes Principales y las Características Originales (PCA)')
plt.legend()
plt.xticks(rotation=90)

# Subgráfica para SVD
plt.subplot(1, 2, 2)
plt.bar(df.columns[:-1], components_svd[0], label='Componente 1')
plt.bar(df.columns[:-1], components_svd[1], label='Componente 2', alpha=0.7)
plt.xlabel('Características')
plt.ylabel('Valor de carga')
plt.title('Relación entre los Componentes Singulares y las Características Originales (SVD)')
plt.legend()
plt.xticks(rotation=90)

# Ajustar el layout para que las gráficas no se solapen
plt.tight_layout()
plt.show()
```



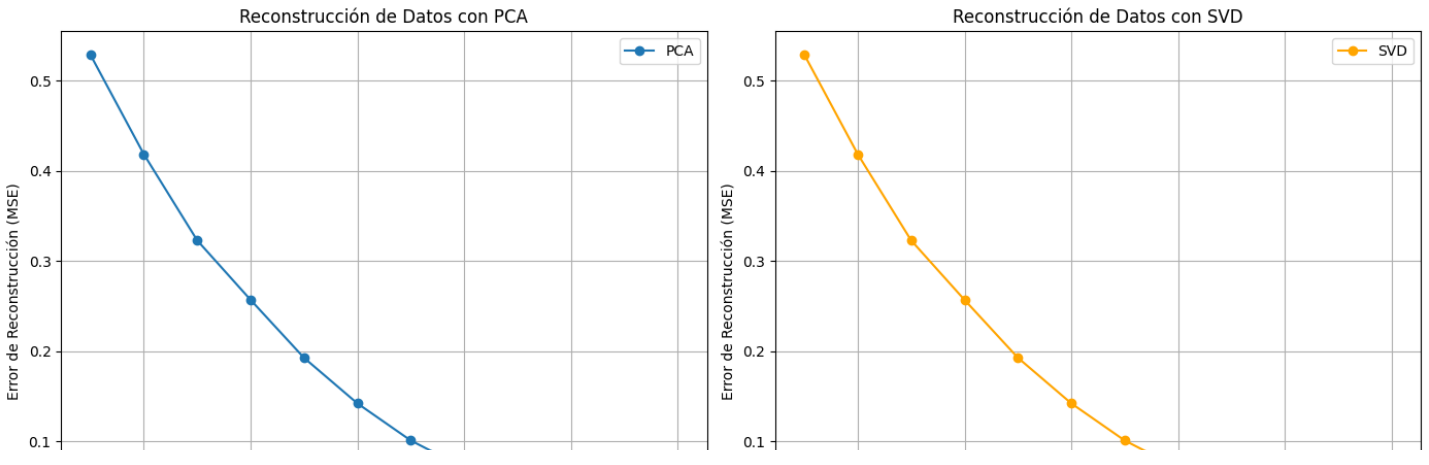

1. **Comparación de la reconstrucción de los datos utilizando diferentes números de componentes en PCA y SVD**

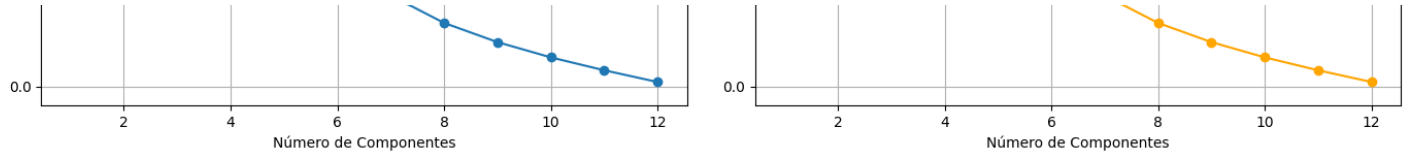
   **El objetivo de esta sección es comparar cómo la precisión de la reconstrucción de los datos varía al utilizar diferentes cantidades de componentes principales en PCA y SVD. Al reducir la dimensionalidad, se pierde información, por lo que es importante evaluar cuánta información se conserva con distintos números de componentes y cómo esto afecta a la reconstrucción de los datos originales.**

   - **PCA: En PCA, los datos se proyectan sobre los primeros n componentes principales para reducir la dimensionalidad. Cuantos más componentes se utilicen, mayor será la precisión de la reconstrucción, ya que se retendrá más varianza de los datos originales. Sin embargo, al utilizar menos componentes, se perderá más información, lo que se reflejará en un mayor error de reconstrucción.**
   - **SVD: Similar a PCA, en SVD, los datos se proyectan sobre los primeros n vectores singulares. Al igual que en PCA, usar más componentes mejora la reconstrucción, mientras que usar menos componentes incrementa el error de reconstrucción. Los valores singulares indican la magnitud de la varianza capturada por cada componente, y los primeros componentes retienen la mayor parte de la información.**

In [10]:

```
from sklearn.metrics import mean_squared_error
import numpy as np
import matplotlib.pyplot as plt

# Función para calcular el error de reconstrucción
def reconstruction_error(original_data, reconstructed_data):
    return mean_squared_error(original_data, reconstructed_data)

# Inicializar listas para guardar errores
pca_errors = []
svd_errors = []
components_range = range(1, min(scaled_data.shape[1], df.shape[1]))

# Realizar la reconstrucción con PCA
for n_components in components_range:
    pca = PCA(n_components=n_components)
    pca_result = pca.fit_transform(scaled_data)
    reconstructed_data_pca = pca.inverse_transform(pca_result)

    # Calcular el error de reconstrucción para PCA
    pca_errors.append(reconstruction_error(scaled_data, reconstructed_data_pca))

# Realizar la reconstrucción con SVD
for n_components in components_range:
    svd = TruncatedSVD(n_components=n_components)
    svd_result = svd.fit_transform(scaled_data)
    reconstructed_data_svd = svd.inverse_transform(svd_result)

    # Calcular el error de reconstrucción para SVD
    svd_errors.append(reconstruction_error(scaled_data, reconstructed_data_svd))

# Crear la figura con dos subgráficas
plt.figure(figsize=(14, 6))

# Subgráfica para PCA
plt.subplot(1, 2, 1)
plt.plot(components_range, pca_errors, label='PCA', marker='o')
plt.xlabel('Número de Componentes')
plt.ylabel('Error de Reconstrucción (MSE)')
plt.title('Reconstrucción de Datos con PCA')
plt.grid(True)
plt.legend()

# Subgráfica para SVD
plt.subplot(1, 2, 2)
plt.plot(components_range, svd_errors, label='SVD', marker='o', color='orange')
plt.xlabel('Número de Componentes')
plt.ylabel('Error de Reconstrucción (MSE)')
plt.title('Reconstrucción de Datos con SVD')
plt.grid(True)
plt.legend()

# Ajustar el layout para que las gráficas no se solapen
plt.tight_layout()
plt.show()
```



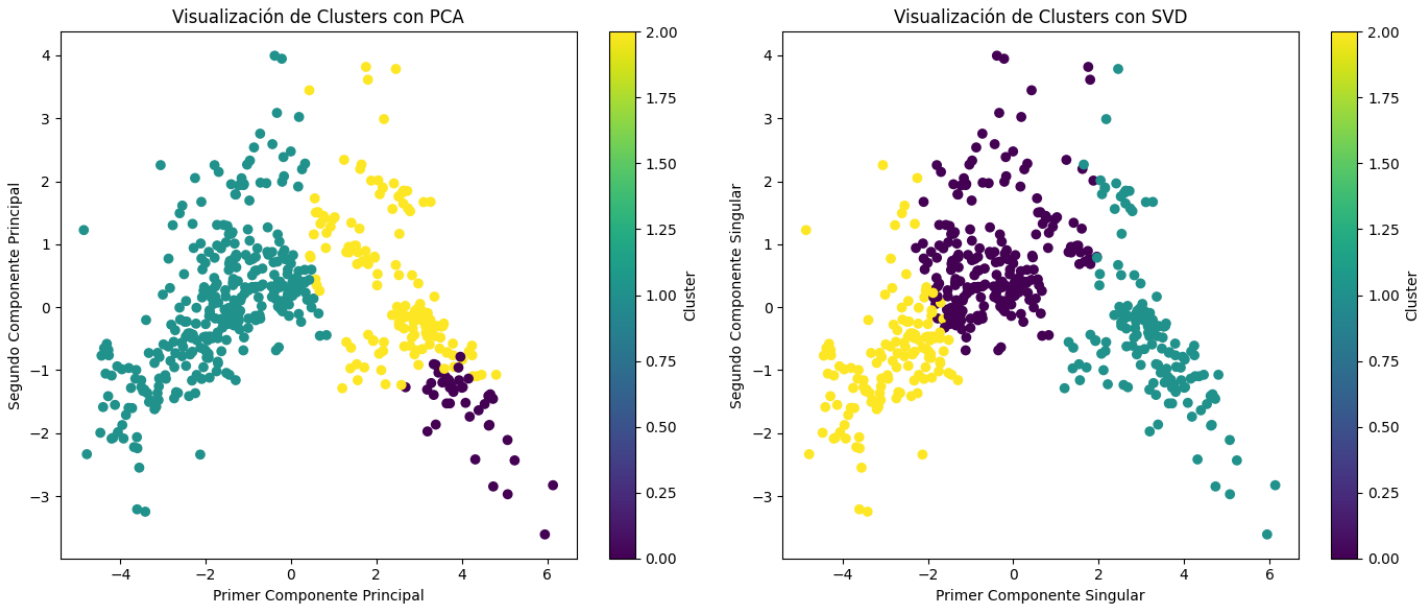



1. **Uso de PCA y SVD para mejorar la visualización de clusters en el conjunto de datos de Boston**

   **El objetivo de esta sección es utilizar PCA y SVD para reducir la dimensionalidad del conjunto de datos de Boston y observar cómo la reducción de la dimensionalidad mejora la separación de los diferentes clusters o grupos en los datos. Para ello, aplicaremos el algoritmo de KMeans para encontrar tres clusters en los datos y visualizaremos cómo estos clusters se distribuyen en el espacio reducido.**

In [11]:

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# K-Means para la visualización de clusters

kmeans_pca = KMeans(n_clusters=3)
df['Cluster_PCA'] = kmeans_pca.fit_predict(pca_result)

kmeans_svd = KMeans(n_clusters=3)
df['Cluster_SVD'] = kmeans_svd.fit_predict(svd_result)

# Crear la figura con dos subgráficas
plt.figure(figsize=(14, 6))

# Subgráfica para PCA
plt.subplot(1, 2, 1)
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=df['Cluster_PCA'], cmap='viridis')
plt.xlabel('Primer Componente Principal')
plt.ylabel('Segundo Componente Principal')
plt.title('Visualización de Clusters con PCA')
plt.colorbar(label='Cluster')

# Subgráfica para SVD
plt.subplot(1, 2, 2)
plt.scatter(svd_result[:, 0], svd_result[:, 1], c=df['Cluster_SVD'], cmap='viridis')
plt.xlabel('Primer Componente Singular')
plt.ylabel('Segundo Componente Singular')
plt.title('Visualización de Clusters con SVD')
plt.colorbar(label='Cluster')

# Ajustar el layout para que las gráficas no se solapen
plt.tight_layout()
plt.show()
```

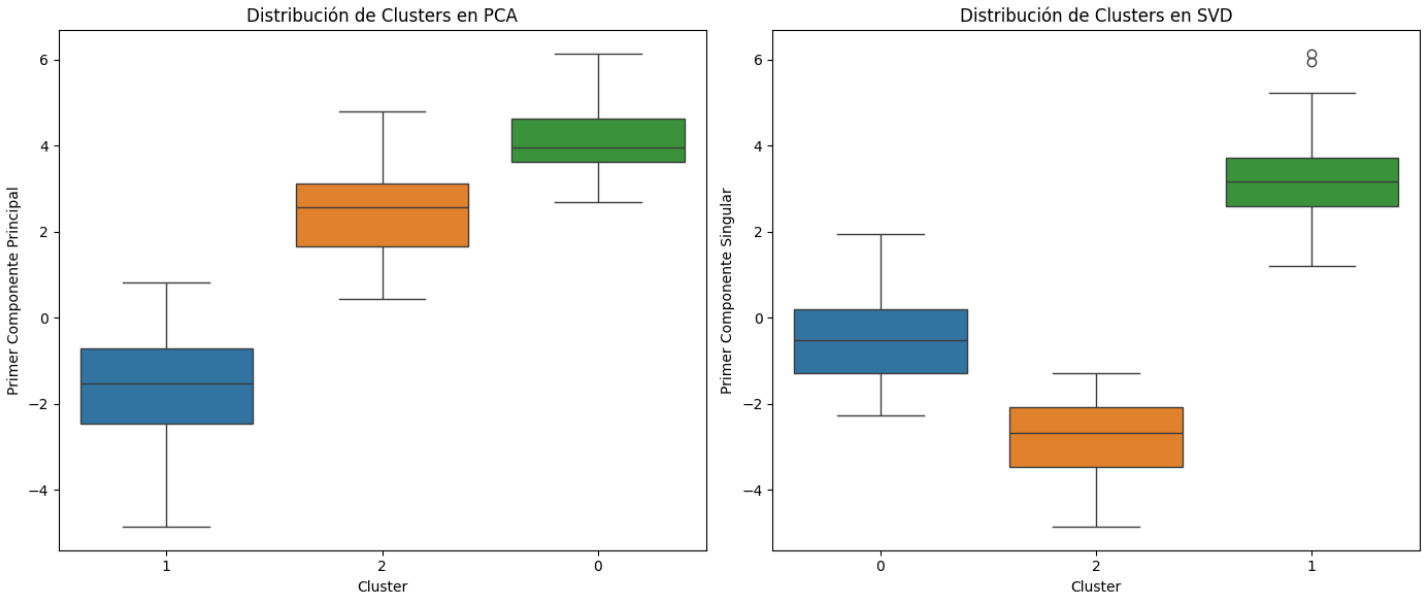


1. **Análisis de la Distribución de los Clusters con Boxplots para PCA y SVD**

   **En este ejercicio, analizaremos la distribución de los valores de los clusters generados a partir de PCA y SVD, utilizando gráficos de Boxplot. Esto permitirá visualizar la dispersión de los valores en cada cluster y evaluar qué tan bien se diferencian en el espacio reducido.**

In [12]:

```
import seaborn as sns


# Asegurarnos de seleccionar solo las dos primeras componentes
pca_df = pd.DataFrame(pca_result[:, :2], columns=['PC1', 'PC2'])
pca_df['Cluster'] = df['Cluster_PCA'].astype(str)  # Convertir a string para evitar prob
lemas en la visualización

svd_df = pd.DataFrame(svd_result[:, :2], columns=['SVD1', 'SVD2'])
svd_df['Cluster'] = df['Cluster_SVD'].astype(str)  # Convertir a string para evitar prob
lemas en la visualización

# Crear la figura con dos subgráficas para Boxplots de PCA y SVD
plt.figure(figsize=(14, 6))

# Boxplot para PCA (Componente Principal 1 por Cluster)
plt.subplot(1, 2, 1)
sns.boxplot(x='Cluster', y='PC1', data=pca_df, hue='Cluster', dodge=False, legend=False)
plt.xlabel('Cluster')
plt.ylabel('Primer Componente Principal')
plt.title('Distribución de Clusters en PCA')

# Boxplot para SVD (Componente Singular 1 por Cluster)
plt.subplot(1, 2, 2)
sns.boxplot(x='Cluster', y='SVD1', data=svd_df, hue='Cluster', dodge=False, legend=False
)
plt.xlabel('Cluster')
plt.ylabel('Primer Componente Singular')
plt.title('Distribución de Clusters en SVD')

# Ajustar el layout
plt.tight_layout()
plt.show()
```




1. **Eliminación de Outliers en los Clusters Generados por PCA y SVD**

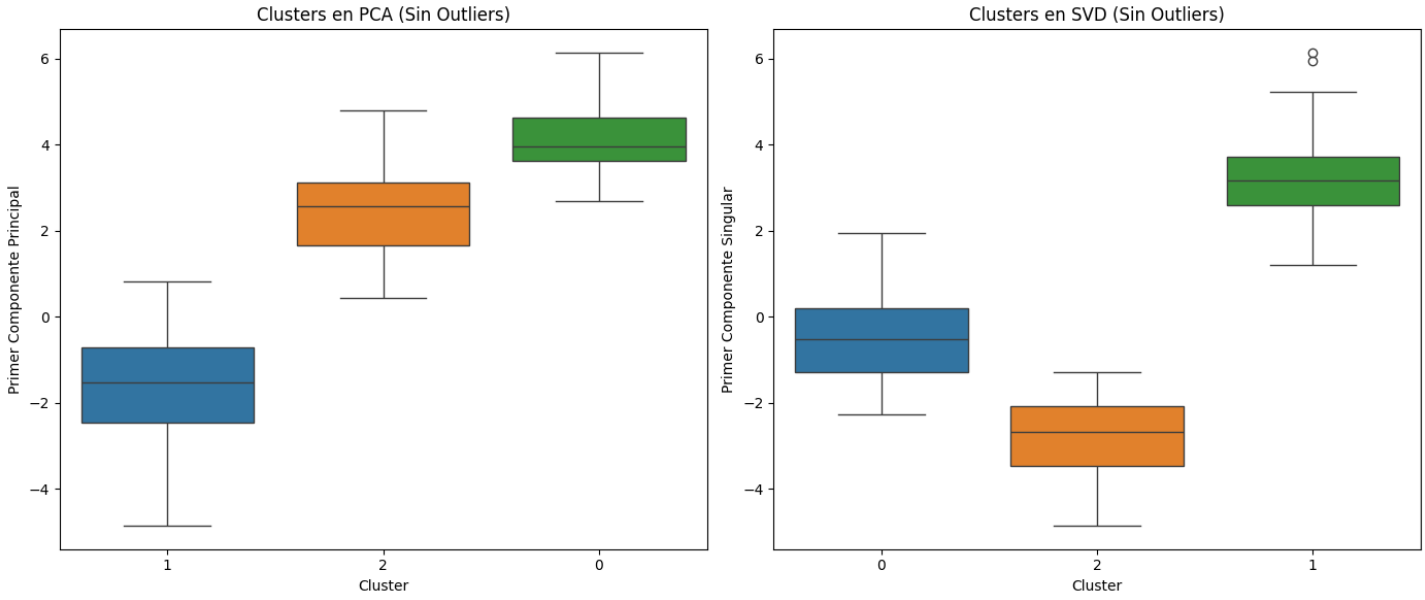
   **En este ejercicio, identificaremos y eliminaremos los outliers dentro de los clusters generados en el análisis con PCA y SVD.**

   **Utilizaremos el rango intercuartílico (IQR) para detectar y filtrar valores atípicos en las primeras componentes de cada técnica. Posteriormente, generaremos boxplots comparativos para visualizar el**

**impacto de la eliminación de outliers.**

In [13]:

```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Función para eliminar outliers usando el rango intercuartílico (IQR)
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Aplicar eliminación de outliers en PCA y SVD
pca_cleaned = remove_outliers(pca_df, 'PC1')
svd_cleaned = remove_outliers(svd_df, 'SVD1')

# Crear la figura con dos subgráficas
plt.figure(figsize=(14, 6))

# Boxplot de PCA sin outliers
plt.subplot(1, 2, 1)
sns.boxplot(x='Cluster', y='PC1', data=pca_cleaned, hue='Cluster', legend=False)
plt.xlabel('Cluster')
plt.ylabel('Primer Componente Principal')
plt.title('Clusters en PCA (Sin Outliers)')

# Boxplot de SVD sin outliers
plt.subplot(1, 2, 2)
sns.boxplot(x='Cluster', y='SVD1', data=svd_cleaned, hue='Cluster', legend=False)
plt.xlabel('Cluster')
plt.ylabel('Primer Componente Singular')
plt.title('Clusters en SVD (Sin Outliers)')

# Ajustar el layout
plt.tight_layout()
plt.show()
```

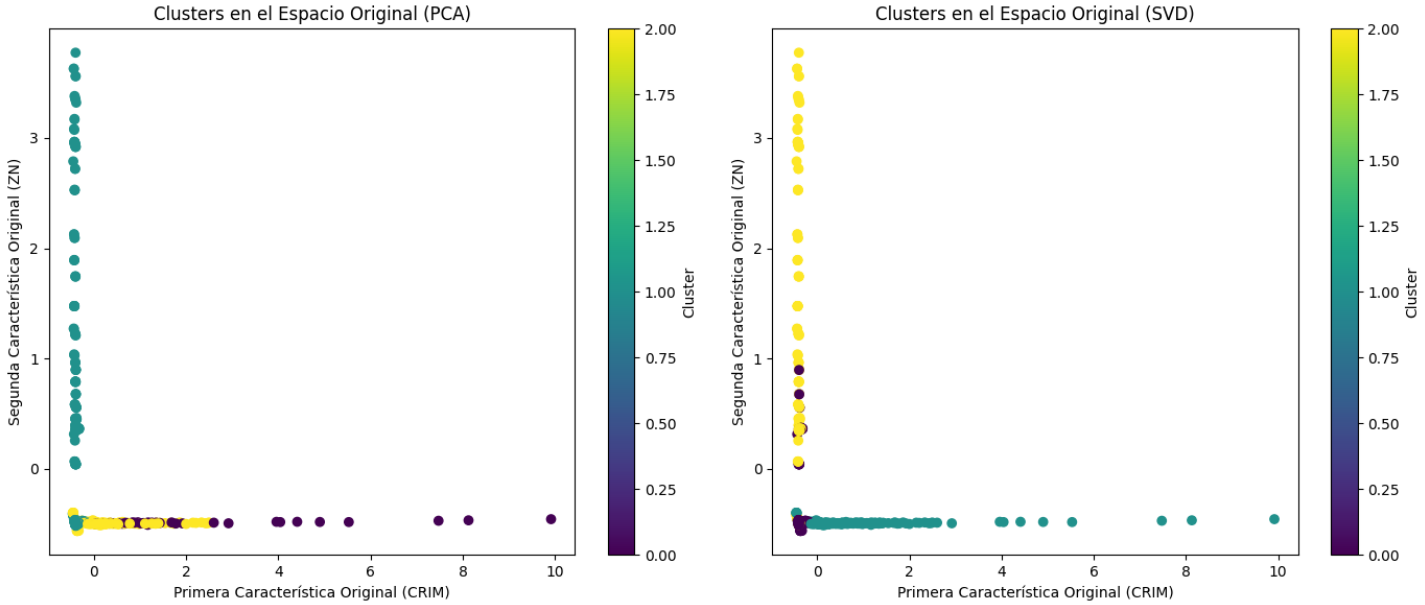



1. **Visualización de los clusters generados en el espacio reducido en el espacio original**

**El objetivo de esta sección es ver cómo los clusters generados a partir de PCA y SVD se proyectan nuevamente al espacio original. Para hacerlo, utilizaremos las transformaciones inversas de PCA y SVD para reconstruir los datos a su forma original y luego visualizaremos cómo se distribuyen los clusters en ese espacio.**

In [14]:

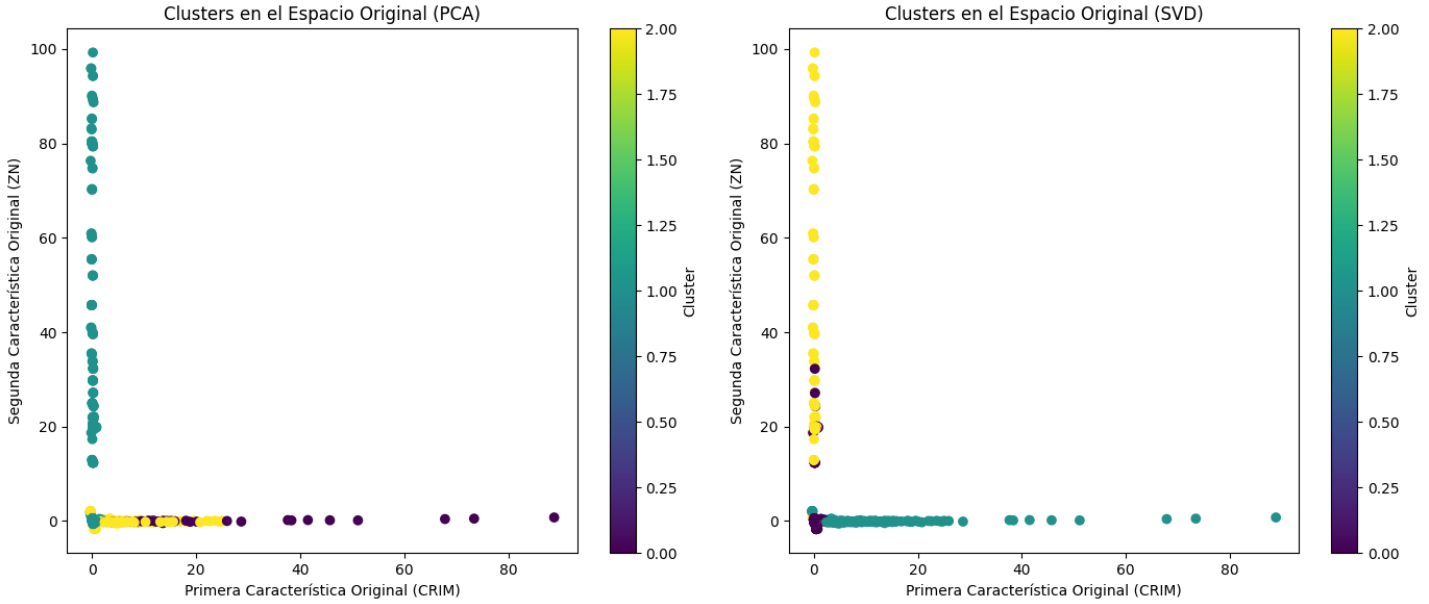
```
# Transformación inversa con PCA
pca_inverse = pca.inverse_transform(pca_result)

# Transformación inversa con SVD
svd_inverse = svd.inverse_transform(svd_result)

# Indices de los campos originales
first_field = 0
second_field = 1

# Visualización de los clusters en el espacio original con PCA
plt.figure(figsize=(14, 6))

# Subgráfica para PCA
plt.subplot(1, 2, 1)
plt.scatter(pca_inverse[:, first_field], pca_inverse[:, second_field], c=df['Cluster_PCA
'], cmap='viridis')
plt.xlabel(f'Primera Característica Original ({df.columns[first_field]})')
plt.ylabel(f'Segunda Característica Original ({df.columns[second_field]})')
plt.title('Clusters en el Espacio Original (PCA)')
plt.colorbar(label='Cluster')

# Subgráfica para SVD
plt.subplot(1, 2, 2)
plt.scatter(svd_inverse[:, first_field], svd_inverse[:, second_field], c=df['Cluster_SVD
'], cmap='viridis')
plt.xlabel(f'Primera Característica Original ({df.columns[first_field]})')
plt.ylabel(f'Segunda Característica Original ({df.columns[second_field]})')
plt.title('Clusters en el Espacio Original (SVD)')
plt.colorbar(label='Cluster')

# Ajustar el layout para que las gráficas no se solapen
plt.tight_layout()
plt.show()
```




In [15]:

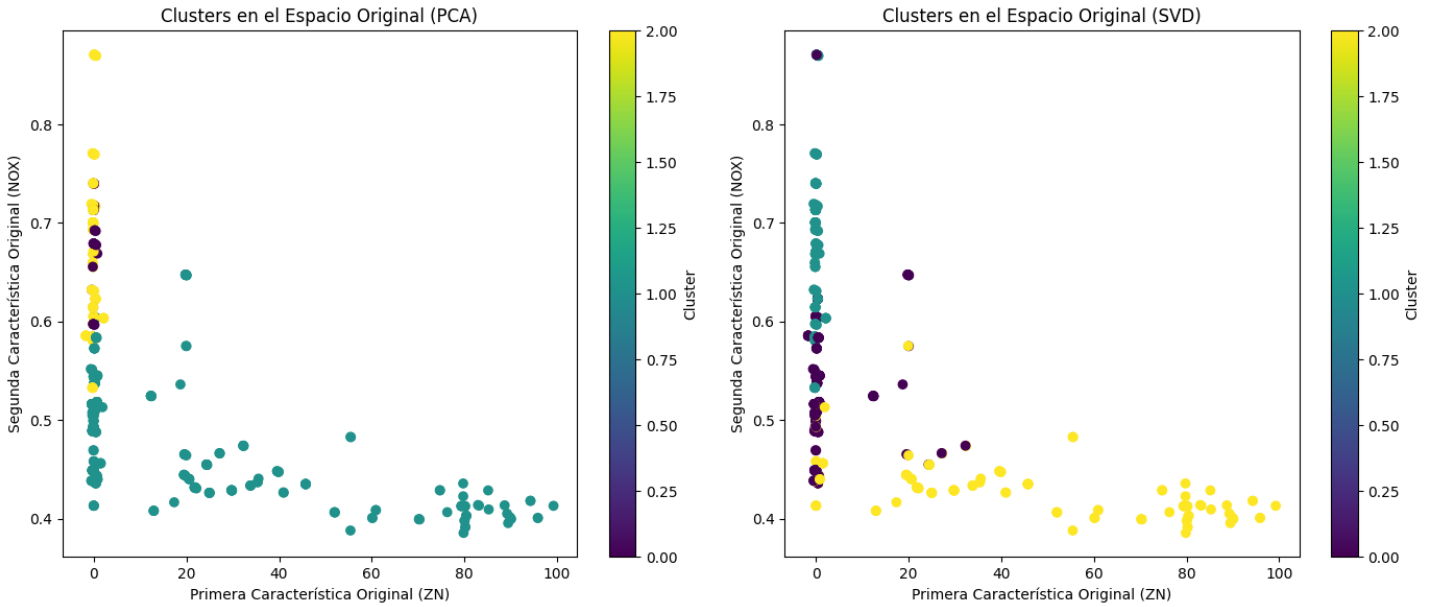
```
# Aplicar la transformación inversa de la estandarización a los datos de PCA y SVD
pca_inverse_original_space = scaler.inverse_transform(pca_inverse)
svd_inverse_original_space = scaler.inverse_transform(svd_inverse)

# Indices de los campos originales
first_field = 0
second_field = 1

# Visualización de los clusters en el espacio original con PCA y SVD
plt.figure(figsize=(14, 6))

# Subgráfica para PCA (con datos en el espacio original)
```

```
plt.subplot(1, 2, 1)
plt.scatter(pca_inverse_original_space[:, first_field], pca_inverse_original_space[:, sec
ond_field], c=df['Cluster_PCA'], cmap='viridis')
plt.xlabel(f'Primera Característica Original ({df.columns[first_field]})')
plt.ylabel(f'Segunda Característica Original ({df.columns[second_field]})')
plt.title('Clusters en el Espacio Original (PCA)')
plt.colorbar(label='Cluster')

# Subgráfica para SVD (con datos en el espacio original)
plt.subplot(1, 2, 2)
plt.scatter(svd_inverse_original_space[:, first_field], svd_inverse_original_space[:, sec
ond_field], c=df['Cluster_SVD'], cmap='viridis')
plt.xlabel(f'Primera Característica Original ({df.columns[first_field]})')
plt.ylabel(f'Segunda Característica Original ({df.columns[second_field]})')
plt.title('Clusters en el Espacio Original (SVD)')
plt.colorbar(label='Cluster')

# Ajustar el layout para que las gráficas no se solapen
plt.tight_layout()
plt.show()
```



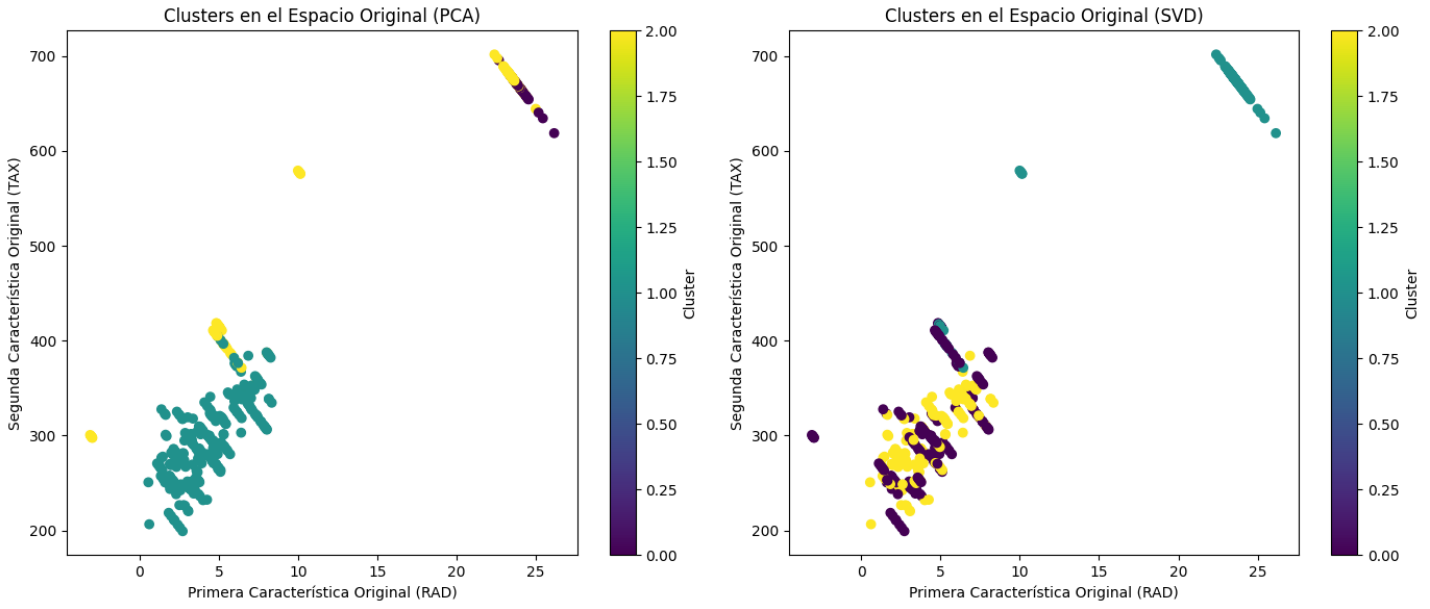

In [16]:

```
# Indices de los campos originales
first_field = 1
second_field = 4

# Visualización de los clusters en el espacio original con PCA y SVD
plt.figure(figsize=(14, 6))

# Subgráfica para PCA (con datos en el espacio original)
plt.subplot(1, 2, 1)
plt.scatter(pca_inverse_original_space[:, first_field], pca_inverse_original_space[:, sec
ond_field], c=df['Cluster_PCA'], cmap='viridis')
plt.xlabel(f'Primera Característica Original ({df.columns[first_field]})')
plt.ylabel(f'Segunda Característica Original ({df.columns[second_field]})')
plt.title('Clusters en el Espacio Original (PCA)')
plt.colorbar(label='Cluster')

# Subgráfica para SVD (con datos en el espacio original)
plt.subplot(1, 2, 2)
plt.scatter(svd_inverse_original_space[:, first_field], svd_inverse_original_space[:, sec
ond_field], c=df['Cluster_SVD'], cmap='viridis')
plt.xlabel(f'Primera Característica Original ({df.columns[first_field]})')
plt.ylabel(f'Segunda Característica Original ({df.columns[second_field]})')
plt.title('Clusters en el Espacio Original (SVD)')
plt.colorbar(label='Cluster')

# Ajustar el layout para que las gráficas no se solapen
plt.tight_layout()
plt.show()
```



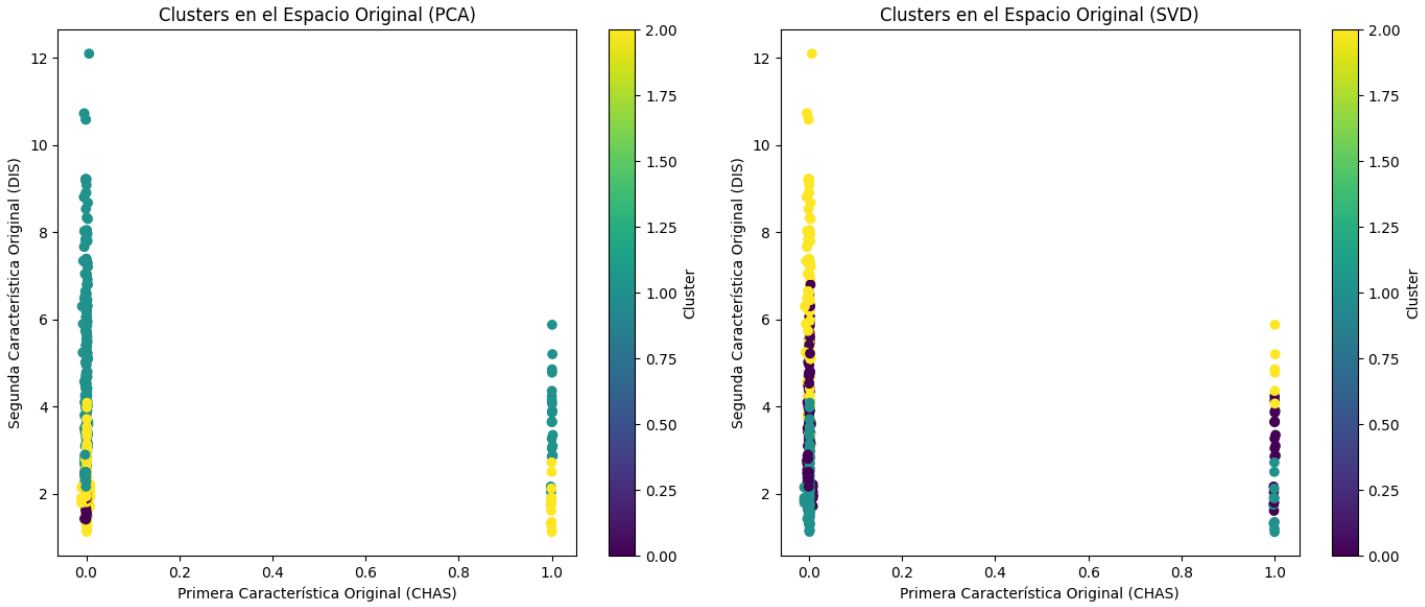

In [17]:

```
# Indices de los campos originales
first_field = 8
second_field = 9

# Visualización de los clusters en el espacio original con PCA y SVD
plt.figure(figsize=(14, 6))

# Subgráfica para PCA (con datos en el espacio original)
plt.subplot(1, 2, 1)
plt.scatter(pca_inverse_original_space[:, first_field], pca_inverse_original_space[:, sec
ond_field], c=df['Cluster_PCA'], cmap='viridis')
plt.xlabel(f'Primera Característica Original ({df.columns[first_field]})')
plt.ylabel(f'Segunda Característica Original ({df.columns[second_field]})')
plt.title('Clusters en el Espacio Original (PCA)')
plt.colorbar(label='Cluster')

# Subgráfica para SVD (con datos en el espacio original)
plt.subplot(1, 2, 2)
plt.scatter(svd_inverse_original_space[:, first_field], svd_inverse_original_space[:, sec
ond_field], c=df['Cluster_SVD'], cmap='viridis')
plt.xlabel(f'Primera Característica Original ({df.columns[first_field]})')
plt.ylabel(f'Segunda Característica Original ({df.columns[second_field]})')
plt.title('Clusters en el Espacio Original (SVD)')
plt.colorbar(label='Cluster')

# Ajustar el layout para que las gráficas no se solapen
plt.tight_layout()
plt.show()
```



In [18]:

```
# Indices de los campos originales
first_field = 3
second_field = 7

# Visualización de los clusters en el espacio original con PCA y SVD
plt.figure(figsize=(14, 6))

# Subgráfica para PCA (con datos en el espacio original)
plt.subplot(1, 2, 1)
plt.scatter(pca_inverse_original_space[:, first_field], pca_inverse_original_space[:, sec
ond_field], c=df['Cluster_PCA'], cmap='viridis')
plt.xlabel(f'Primera Característica Original ({df.columns[first_field]})')
plt.ylabel(f'Segunda Característica Original ({df.columns[second_field]})')
plt.title('Clusters en el Espacio Original (PCA)')
plt.colorbar(label='Cluster')

# Subgráfica para SVD (con datos en el espacio original)
plt.subplot(1, 2, 2)
plt.scatter(svd_inverse_original_space[:, first_field], svd_inverse_original_space[:, sec
ond_field], c=df['Cluster_SVD'], cmap='viridis')
plt.xlabel(f'Primera Característica Original ({df.columns[first_field]})')
plt.ylabel(f'Segunda Característica Original ({df.columns[second_field]})')
plt.title('Clusters en el Espacio Original (SVD)')
plt.colorbar(label='Cluster')

# Ajustar el layout para que las gráficas no se solapen
plt.tight_layout()
plt.show()
```




In [19]:

```
# Indices de los campos originales
first_field = 6
second_field = 7

# Visualización de los clusters en el espacio original con PCA y SVD
plt.figure(figsize=(14, 6))

# Subgráfica para PCA (con datos en el espacio original)
plt.subplot(1, 2, 1)
plt.scatter(pca_inverse_original_space[:, first_field], pca_inverse_original_space[:, sec
ond_field], c=df['Cluster_PCA'], cmap='viridis')
plt.xlabel(f'Primera Característica Original ({df.columns[first_field]})')
plt.ylabel(f'Segunda Característica Original ({df.columns[second_field]})')
plt.title('Clusters en el Espacio Original (PCA)')
plt.colorbar(label='Cluster')
```

```
# Subgráfica para SVD (con datos en el espacio original)
plt.subplot(1, 2, 2)
plt.scatter(svd_inverse_original_space[:, first_field], svd_inverse_original_space[:, sec
ond_field], c=df['Cluster_SVD'], cmap='viridis')
plt.xlabel(f'Primera Característica Original ({df.columns[first_field]})')
plt.ylabel(f'Segunda Característica Original ({df.columns[second_field]})')
plt.title('Clusters en el Espacio Original (SVD)')
plt.colorbar(label='Cluster')

# Ajustar el layout para que las gráficas no se solapen
plt.tight_layout()
plt.show()
```