In [ ]:

INFORME

CLEANING BANK MARKETING CAMPAIGN DATA

Noviembre 13, 2024

1.1 1. Importar paquetes

In [14]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
from sklearn.decomposition import PCA
```

1.2 2. Cargar los datos

In [15]:
```python
df_economics = pd.read_csv('economics.csv')
df_client = pd.read_csv('client.csv')
df_campaign = pd.read_csv('campaign.csv')
```

In [ ]:

1.3 3. Mostramos informacion de cada Data Frame

In [16]:
```python
df_economics.info(),df_client.info(),df_campaign.info()
```

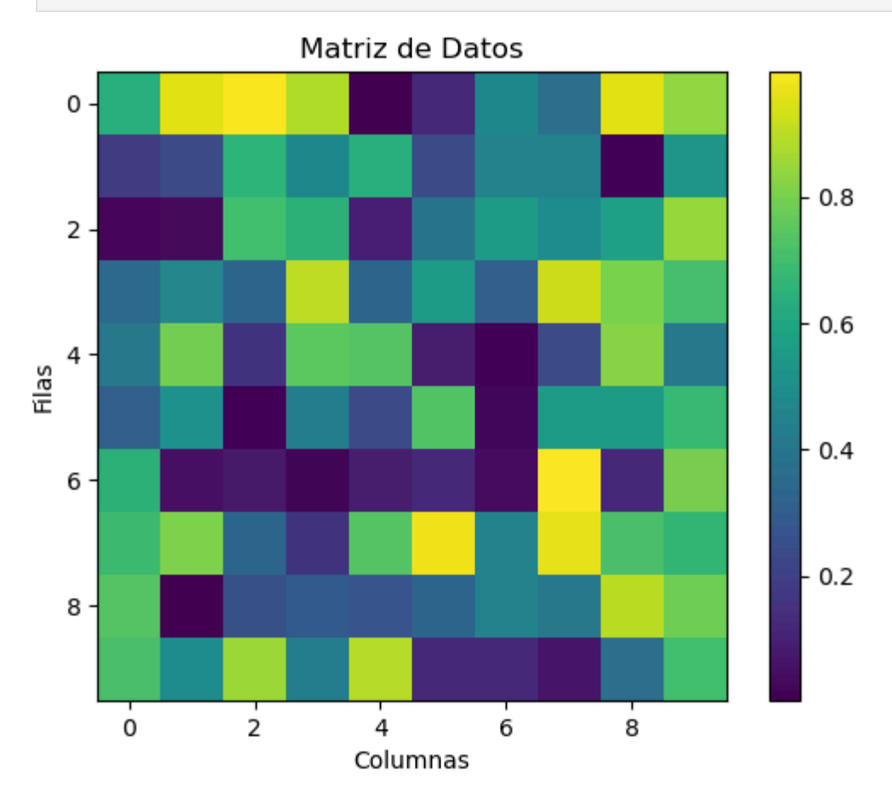
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   client_id             41188 non-null  int64
 1   cons_price_idx        41188 non-null  float64
 2   euribor_three_months  41188 non-null  float64
dtypes: float64(2), int64(1)
memory usage: 965.5 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   client_id       41188 non-null  int64
 1   age             41188 non-null  int64
 2   job             41188 non-null  object
 3   marital         41188 non-null  object
 4   education       39457 non-null  object
 5   credit_default  41188 non-null  bool
 6   mortgage        41188 non-null  bool
dtypes: bool(2), int64(2), object(3)
memory usage: 1.6+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 7 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   client_id                   41188 non-null  int64
 1   number_contacts             41188 non-null  int64
 2   contact_duration            41188 non-null  int64
 3   previous_campaign_contacts  41188 non-null  int64
 4   previous_outcome            41188 non-null  bool
 5   campaign_outcome            41188 non-null  bool
 6   last_contact_date           41188 non-null  object
dtypes: bool(2), int64(4), object(1)
memory usage: 1.6+ MB
```

Out[16]: (None, None, None)

2.1.1 Creamos un analisis Grafico

¿ La Matriz de datos nos arroja informacion relevante ?

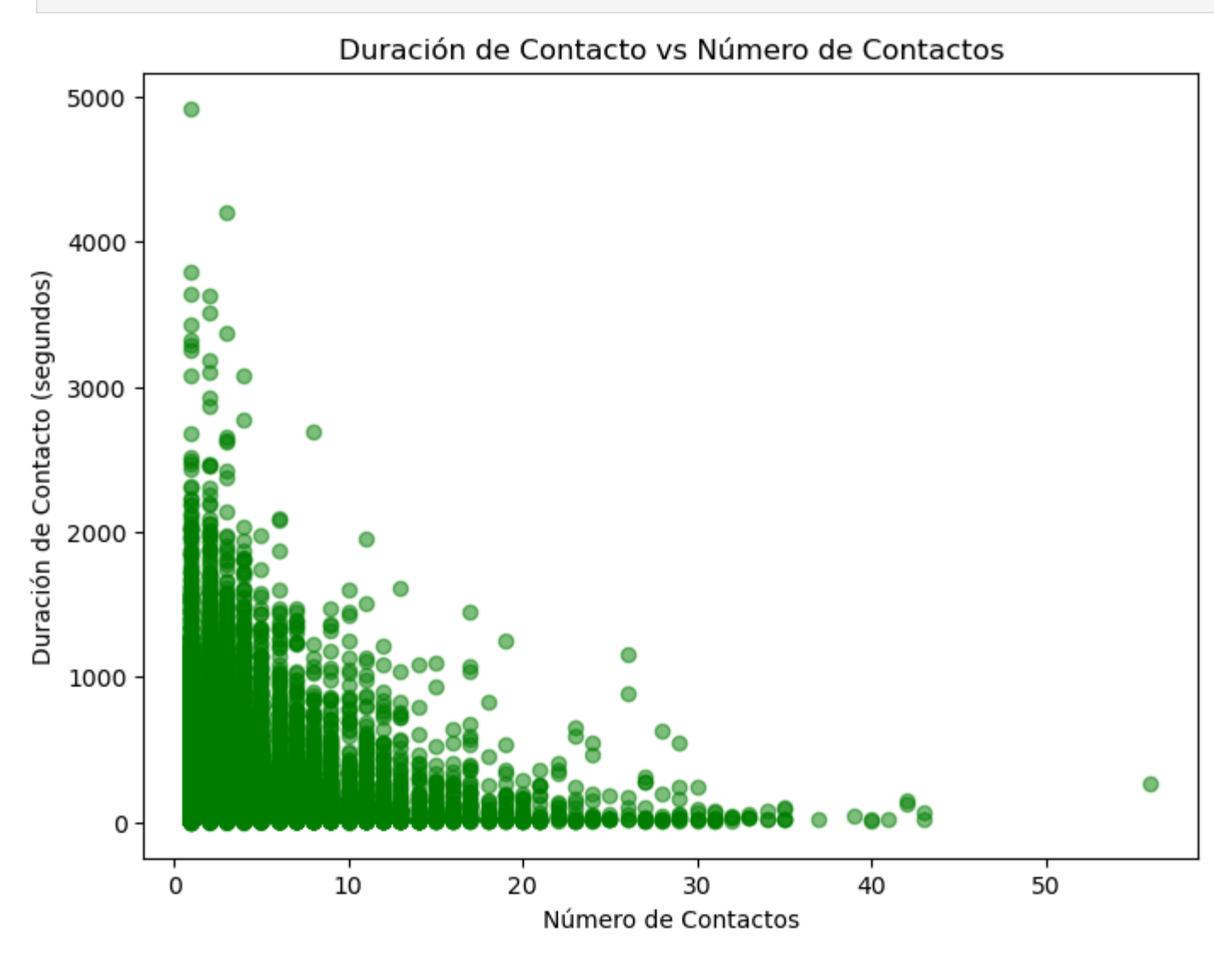
In [35]:
```python
df_campaign = np.random.rand(10, 10)  # Matriz de 10x10 con valores aleatorios entre 0 y 1

# Generar la gráfica
plt.imshow(df_campaign, cmap='viridis', interpolation='none')
plt.colorbar()  # Agregar una barra de colores
plt.title("Matriz de Datos")
plt.xlabel("Columnas")
plt.ylabel("Filas")
plt.show()
```



No logre identificar alguna informacion relevante asociada a la matriz generada del dataframe de la campaña

¿ Que cantidad de contactos hubieron en relacion a la duracion (segundos) ?

In [14]:
```python
plt.figure(figsize=(8, 6))
plt.scatter(df_campaign['number_contacts'], df_campaign['contact_duration'], color='green', alpha=0.5)
plt.title("Duración de Contacto vs Número de Contactos")
plt.xlabel("Número de Contactos")
plt.ylabel("Duración de Contacto (segundos)")
plt.show()
```



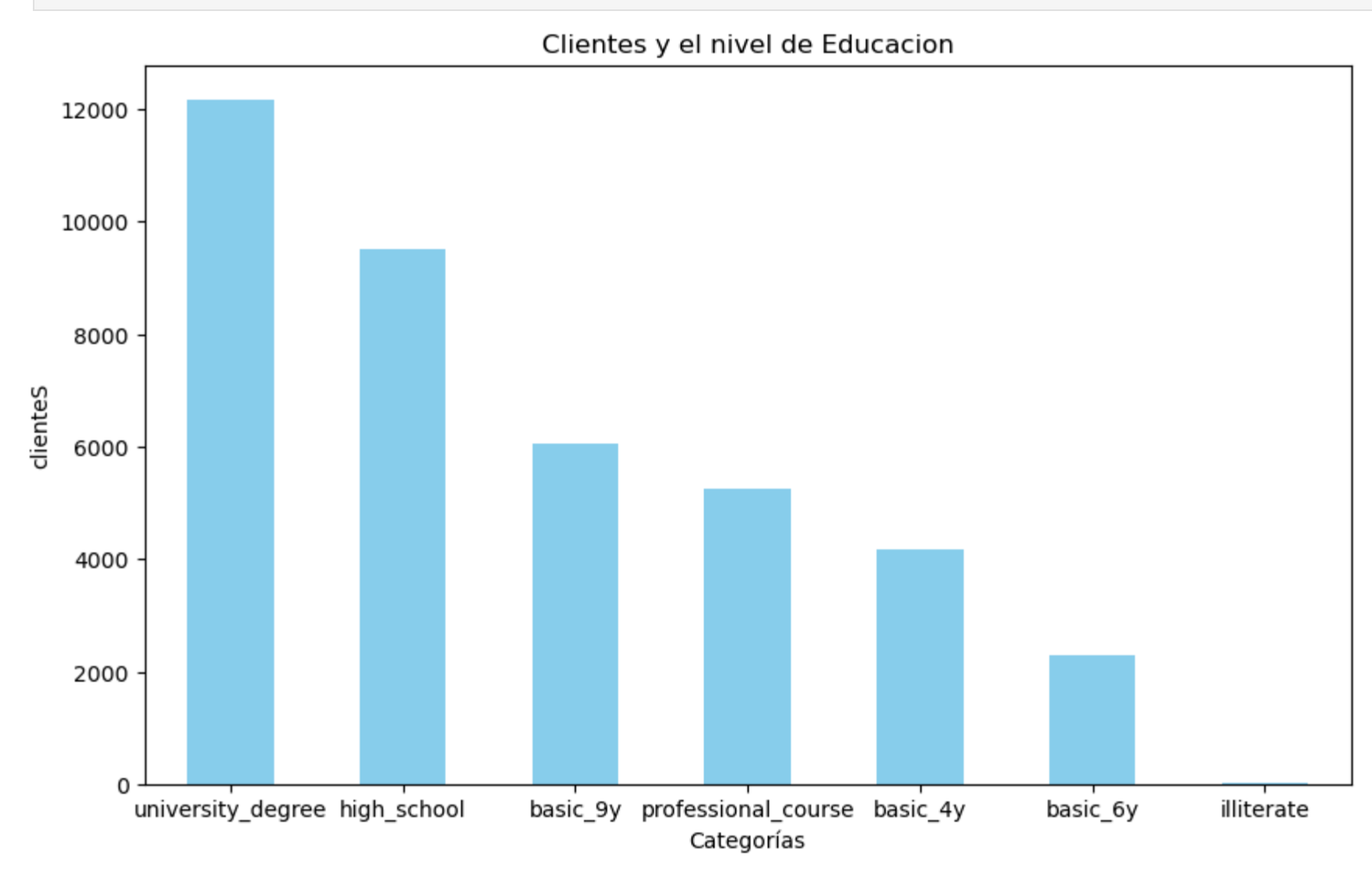
Como podemos identificar la mayor concentracion de contactos que se realizaron fueron entre 0 a 10 contactos y la duracion se concentro entre los 0 y los 2000 segundos

In [ ]:

In [ ]:

¿ Existe alguna tendencia en los clientes por su nivel educacional ?

In [56]:
```python
frequencies = df_client['education'].value_counts()

# Crear gráfico de barras
plt.figure(figsize=(10, 6))
frequencies.plot(kind='bar', color='skyblue')

# Configurar etiquetas
plt.title('Clientes y el nivel de Educacion')
plt.xlabel('Categorías')
plt.ylabel('clienteS')
plt.xticks(rotation=0)  # Para que las categorías no se roten

# Mostrar el gráfico
plt.show()
```



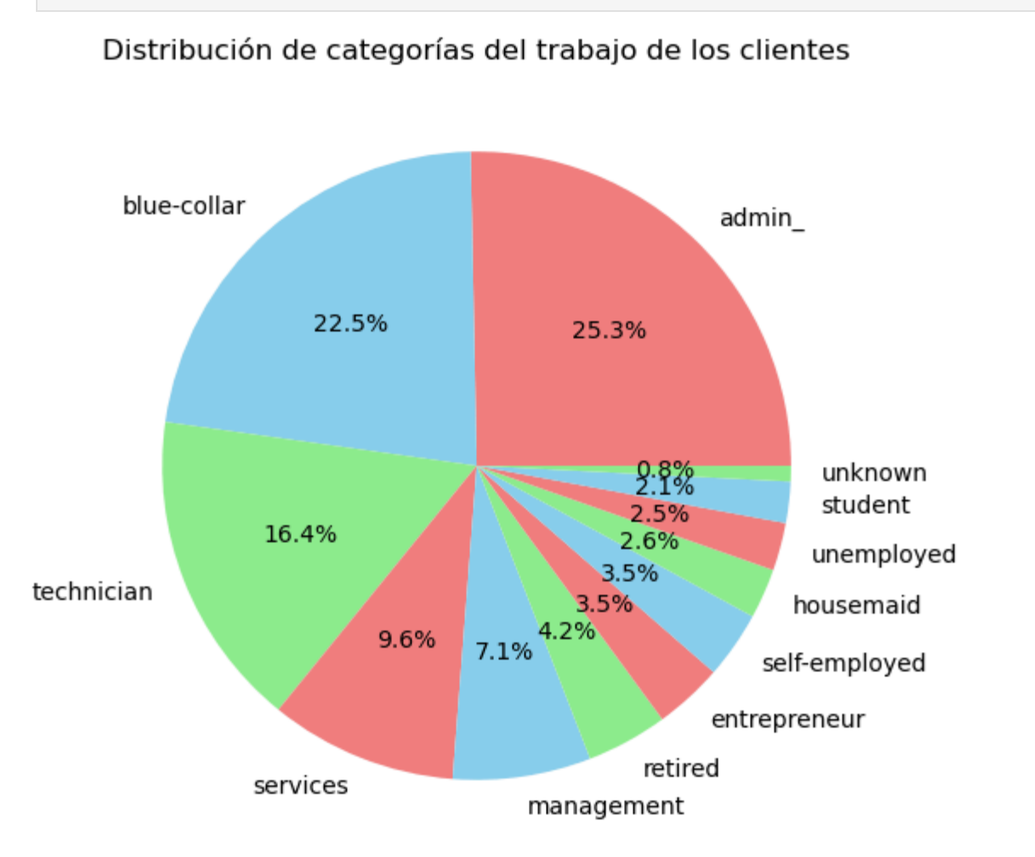
Podemos comprobar que la gran mayoria de los clientes tienen un titulo universitario y que a menor nivel educacional menores clientes

In [ ]:

In [ ]:

¿ Cuantas areas del trabajo se concentran la mayoria de clientes ?

In [60]:
```python
frequencies1 = df_client['job'].value_counts()

plt.figure(figsize=(12, 6))
frequencies1.plot(kind='pie', autopct='%1.1f%%', colors=['lightcoral', 'skyblue', 'lightgreen'])

# Configurar el título
plt.title('Distribución de categorías del trabajo de los clientes')

# Mostrar el gráfico
plt.ylabel('')  # Eliminar la etiqueta del eje Y
plt.show()
```

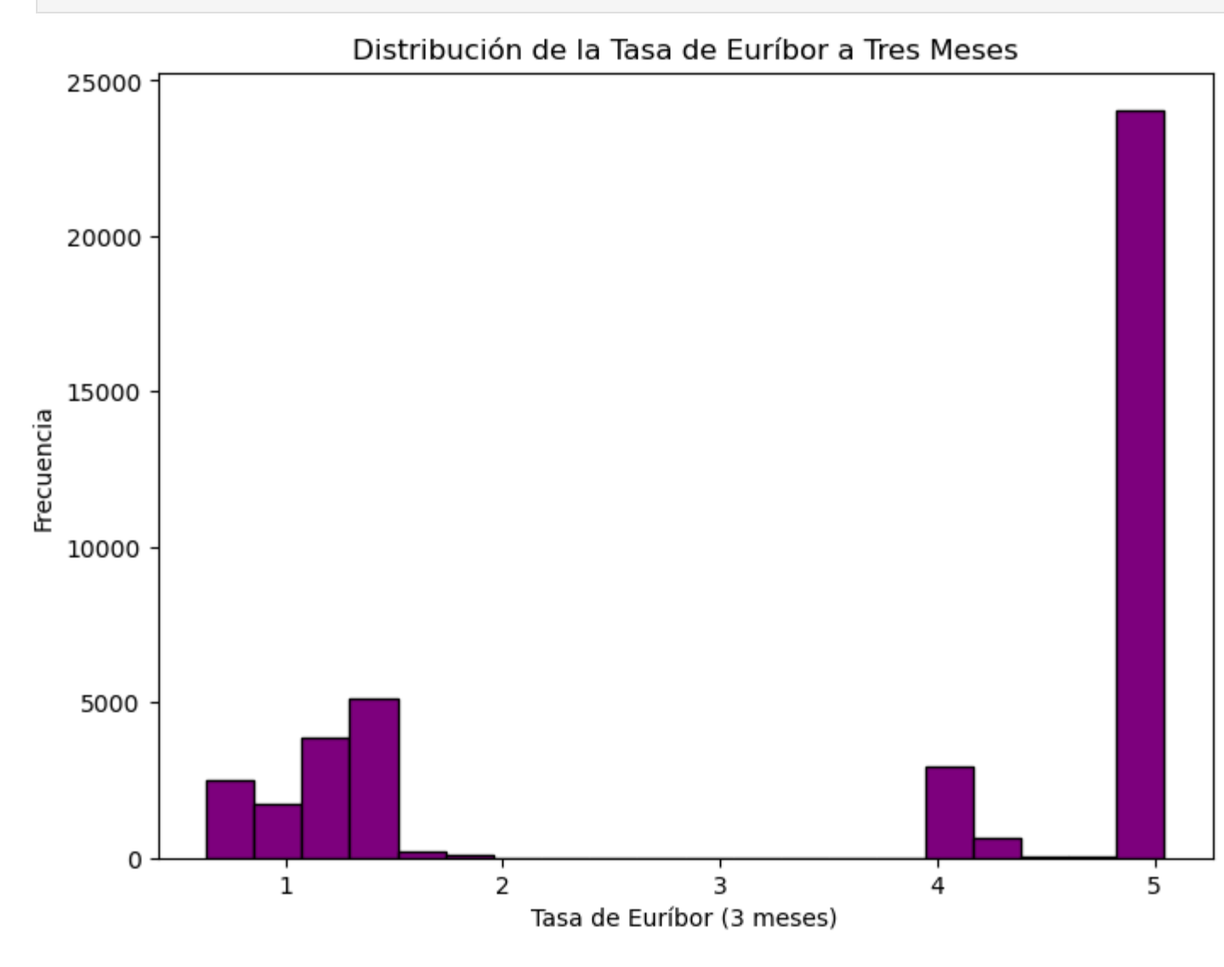


Podemos deducir que la mayor cantidad de trabajadores que solicitaron prestamos se concentran en admin,blue collar y tecnicos en cambio la menor cantidad se concentran los estudiantes

In [ ]:

In [ ]:

In [71]:
```python
# Histograma de la tasa de euríbor a tres meses
plt.figure(figsize=(8, 6))
plt.hist(df_economics['euribor_three_months'], bins=20, color='purple', edgecolor='black')
plt.title("Distribución de la Tasa de Euríbor a Tres Meses")
plt.xlabel("Tasa de Euríbor (3 meses)")
plt.ylabel("Frecuencia")
plt.show()
```



Forma de la distribución: Observa la forma de la distribución; si es simétrica, sesgada a la derecha o a la izquierda. Por ejemplo, si la mayoría de los valores están en el lado izquierdo, indica una asimetría negativa (sesgo hacia valores bajos); si están en el lado derecho, muestra una asimetría positiva.

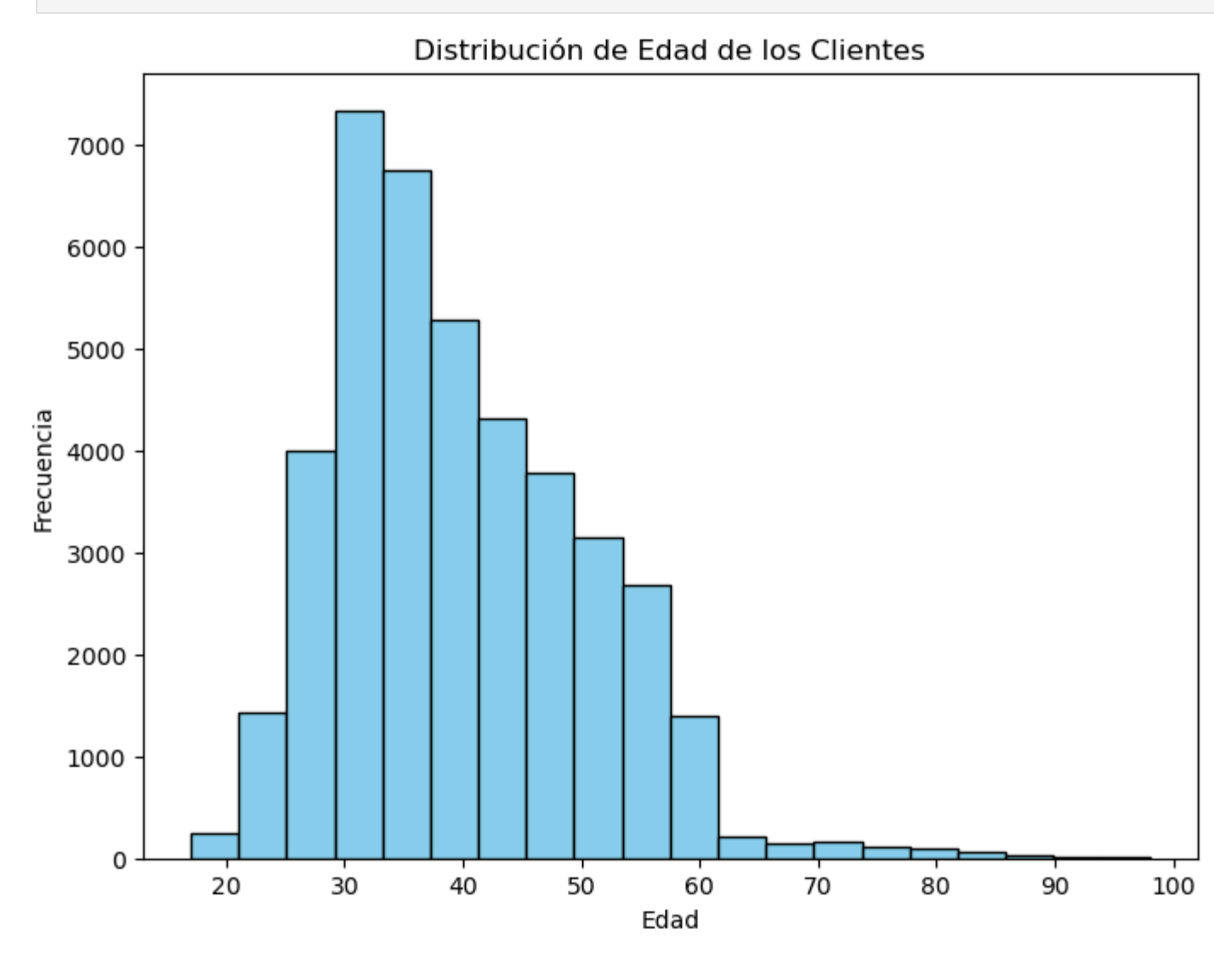
Rango de tasas: Identifica el rango en el eje x para ver cómo varía la tasa de Euríbor. Esto puede indicar en qué intervalos de valores es más probable que se encuentren las tasas.

Frecuencia de valores específicos: Observa los picos para identificar las tasas de Euríbor más comunes durante el periodo de observación

In [ ]:

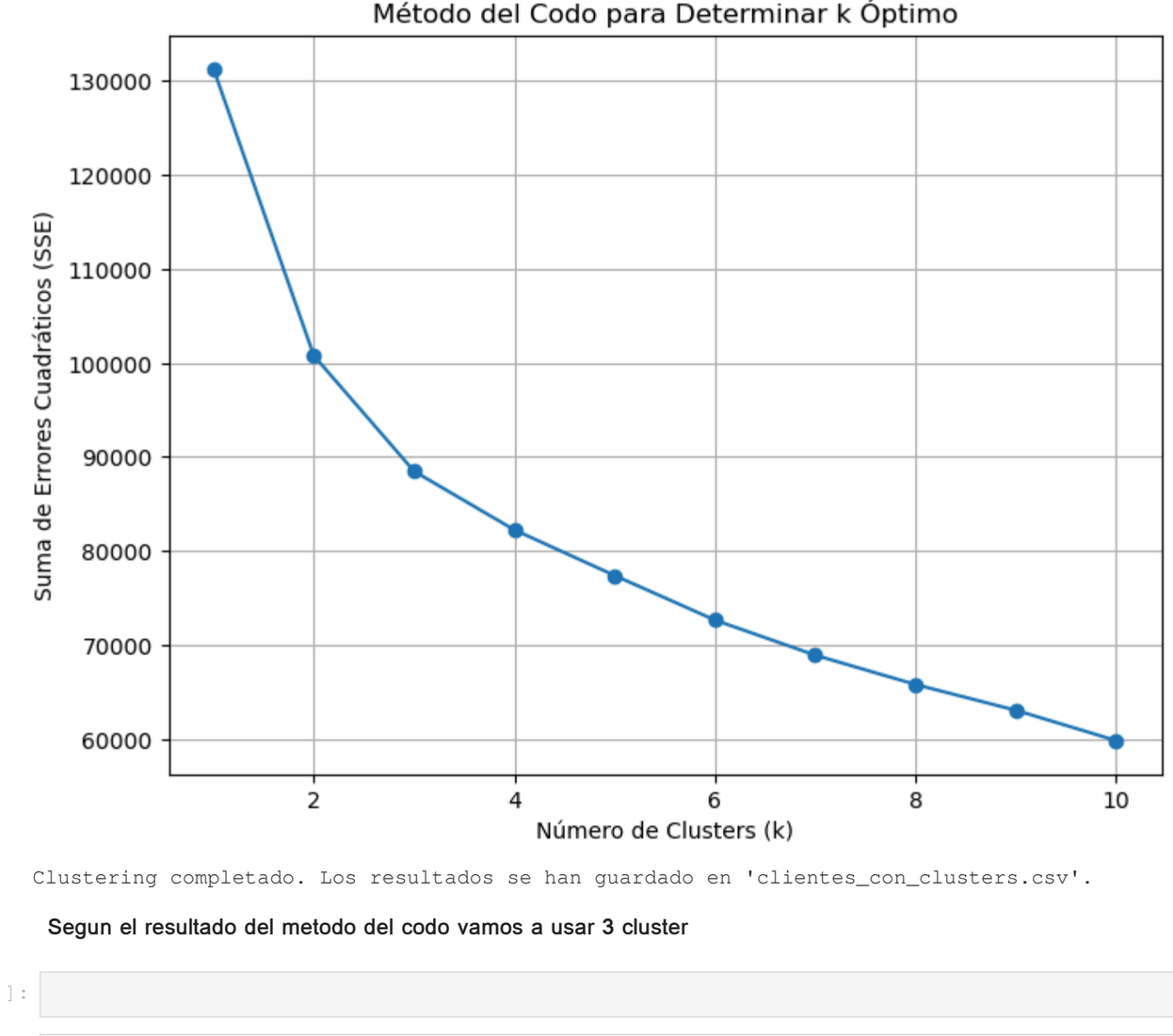
¿ La edad es un factor para solicitar prestamos ?

In [17]:
```python
# Histograma de la distribución de edad
plt.figure(figsize=(8, 6))
plt.hist(df_client['age'], bins=20, color='skyblue', edgecolor='black')
plt.title("Distribución de Edad de los Clientes")
plt.xlabel("Edad")
plt.ylabel("Frecuencia")
plt.show()
```



Los solicitantes entre 29 y 45 concentar la mayor cantidad de prestamos en cambio disminuye esta tendencia despues de los 63 años

In [17]:
```python
# 2. Eliminar columnas irrelevantes
df_client = df_client.drop(columns=['client_id'])

# 3. Preparar variables para codificación y escalamiento
categorical_columns = ['job', 'marital', 'education']
numerical_columns = ['age']

# 4. Codificar variables categóricas y escalar numéricas
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_columns),
        ('cat', OneHotEncoder(), categorical_columns)

    ]
)

# Transformar los datos
X = preprocessor.fit_transform(df_client)

# 5. Método del codo
sse = []
k_range = range(1, 11)  # Probar entre 1 y 10 clusters
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10, max_iter=300)
    kmeans.fit(X)
    sse.append(kmeans.inertia_)

# Graficar el método del codo
plt.figure(figsize=(8, 6))
plt.plot(k_range, sse, marker='o')
plt.xlabel('Número de Clusters (k)')
plt.ylabel('Suma de Errores Cuadráticos (SSE)')
plt.title('Método del Codo para Determinar k Óptimo')
plt.grid()
plt.show()

# 6. Aplicar K-means con el número óptimo de clusters (por ejemplo, 4)
k_optimal = 4  # Ajusta este valor según el gráfico del método del codo
kmeans_final = KMeans(n_clusters=k_optimal, random_state=42, n_init=10, max_iter=300)
clusters = kmeans_final.fit_predict(X)

# 7. Añadir resultados al DataFrame original
df_client['cluster'] = clusters


print("Clustering completado. Los resultados se han guardado en 'clientes_con_clusters.csv'.")
```

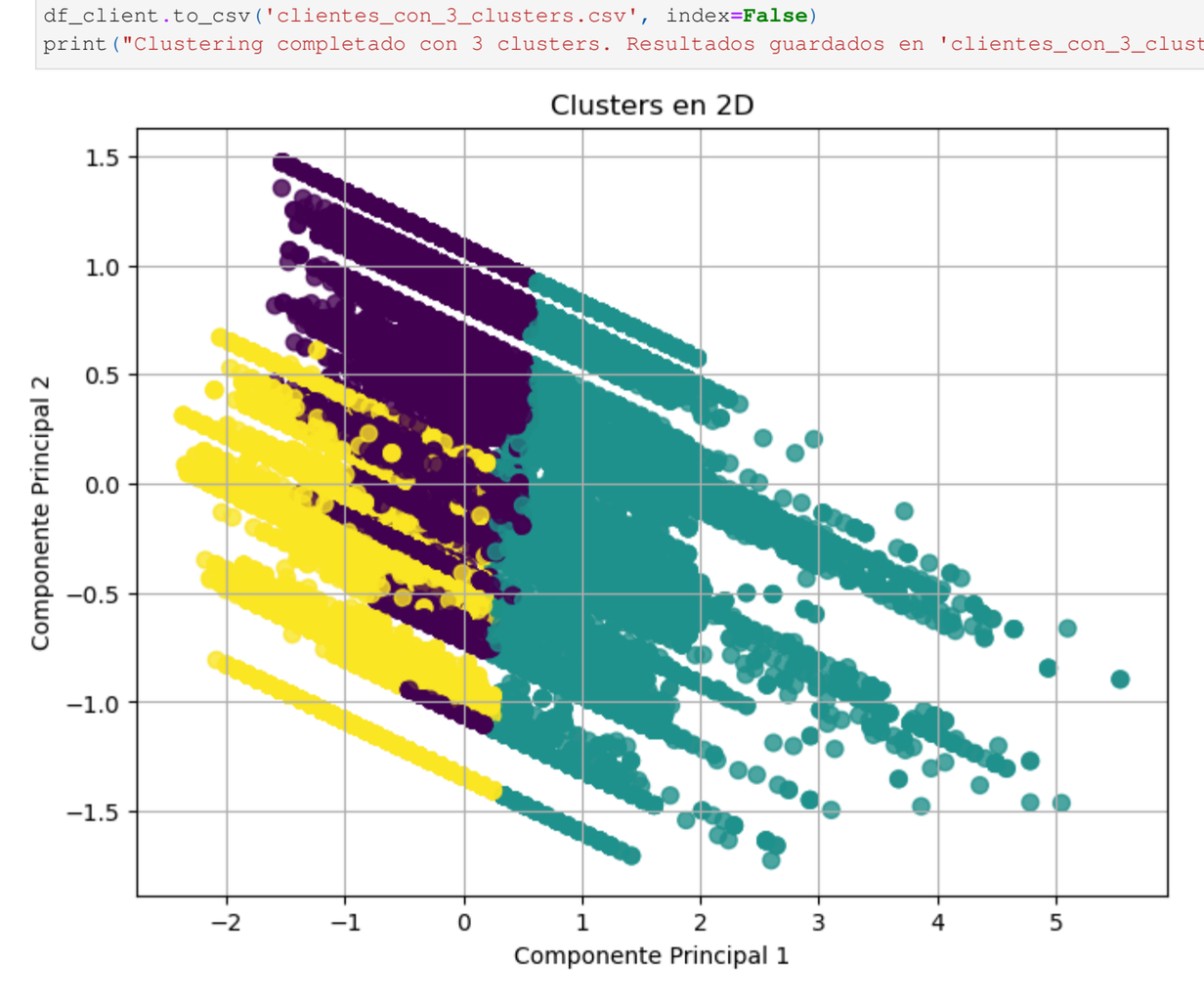


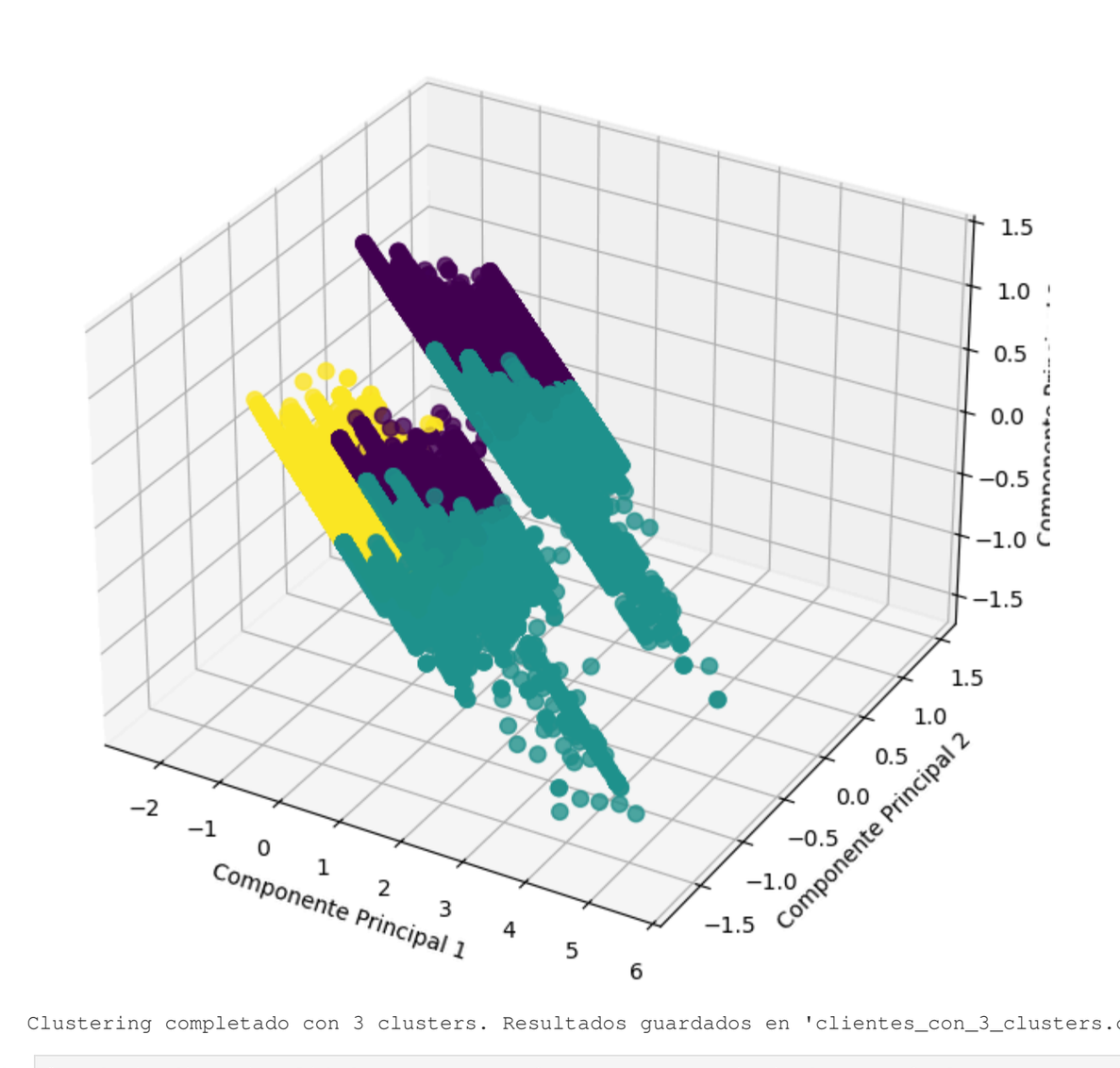

```
Clustering completado. Los resultados se han guardado en 'clientes_con_clusters.csv'.
```

Segun el resultado del metodo del codo vamos a usar 3 cluster

In [ ]:

In [18]:
```python
# 1. Generar 3 clusters
k_optimal = 3  # Cambiado para usar 3 clusters
kmeans_final = KMeans(n_clusters=k_optimal, random_state=42, n_init=10, max_iter=300)
clusters = kmeans_final.fit_predict(X)

# 2. Añadir resultados al DataFrame original
df_client['cluster'] = clusters

# 3. Reducir la dimensionalidad para graficar (PCA a 2D y 3D)
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X)

# Graficar en 2D
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters, cmap='viridis', s=50, alpha=0.8)
plt.title('Clusters en 2D')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.grid()
plt.show()

# Graficar en 3D
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X_pca[:, 0], X_pca[:, 1], X_pca[:, 2], c=clusters, cmap='viridis', s=50, alpha=0.8)
ax.set_title('Clusters en 3D')
ax.set_xlabel('Componente Principal 1')
ax.set_ylabel('Componente Principal 2')
ax.set_zlabel('Componente Principal 3')
plt.show()

# Exportar los resultados
df_client.to_csv('clientes_con_3_clusters.csv', index=False)
print("Clustering completado con 3 clusters. Resultados guardados en 'clientes_con_3_clusters.csv'.")
```






```
Clustering completado con 3 clusters. Resultados guardados en 'clientes_con_3_clusters.csv'.
```

In [20]:
```python
# Reducir dimensionalidad con PCA para 2 componentes principales con 3 clusters
pca = PCA(n_components=2)
X_pca_3 = pca.fit_transform(X)

# Aplicar KMeans con 3 clusters
kmeans_3 = KMeans(n_clusters=3, random_state=42, n_init=10, max_iter=300)
clusters_3 = kmeans_3.fit_predict(X)

# Crear un DataFrame para visualización con PCA y clusters
df_viz_3 = pd.DataFrame(data=X_pca_3, columns=['PC1', 'PC2'])
df_viz_3['cluster'] = clusters_3

# Visualizar los clusters en un gráfico de dispersión
plt.figure(figsize=(10, 7))
for cluster in df_viz_3['cluster'].unique():
    cluster_data = df_viz_3[df_viz_3['cluster'] == cluster]
    plt.scatter(cluster_data['PC1'], cluster_data['PC2'], label=f'Cluster {cluster}')

plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.title('Visualización de Clusters con PCA (k=3)')
plt.legend()
plt.grid()
plt.show()
```