In [1]:
```
!pip install pandas numpy scikit-learn matplotlib seaborn
```

```
Requirement already satisfied: pandas in c:\python\python311\lib\site-packages (2.0.3)
Requirement already satisfied: numpy in c:\python\python311\lib\site-packages (1.25.2)
Requirement already satisfied: scikit-learn in c:\python\python311\lib\site-packages (1.5.1)
Requirement already satisfied: matplotlib in c:\python\python311\lib\site-packages (3.7.2)
Requirement already satisfied: seaborn in c:\python\python311\lib\site-packages (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\python\python311\lib\site-packages (from pandas) (2.
8.2)
Requirement already satisfied: pytz>=2020.1 in c:\python\python311\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in c:\python\python311\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: scipy>=1.6.0 in c:\python\python311\lib\site-packages (from scikit-learn) (1.14.0
)
Requirement already satisfied: joblib>=1.2.0 in c:\python\python311\lib\site-packages (from scikit-learn) (1.4.2
)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\python\python311\lib\site-packages (from scikit-learn)
(3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\python\python311\lib\site-packages (from matplotlib) (1.1.
0)
Requirement already satisfied: cycler>=0.10 in c:\python\python311\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\python\python311\lib\site-packages (from matplotlib) (4.4
2.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\python\python311\lib\site-packages (from matplotlib) (1.4
.4)
Requirement already satisfied: packaging>=20.0 in c:\python\python311\lib\site-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\python\python311\lib\site-packages (from matplotlib) (10.0.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\python\python311\lib\site-packages (from matplotlib)
(3.0.9)
Requirement already satisfied: six>=1.5 in c:\python\python311\lib\site-packages (from python-dateutil>=2.8.2->p
andas) (1.16.0)
WARNING: Ignoring invalid distribution ~ip (C:\Python\Python311\Lib\site-packages)
WARNING: Ignoring invalid distribution ~ip (C:\Python\Python311\Lib\site-packages)
WARNING: Ignoring invalid distribution ~ip (C:\Python\Python311\Lib\site-packages)
```

# Parte I

In [2]:
```
# Importar las bibliotecas necesarias
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Cargar la base de datos
data = pd.read_csv("corporal.csv")

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

Out[2]:

| | edad | peso | altura | sexo | muneca | biceps |
|---|---|---|---|---|---|---|
| **0** | 43 | 87.3 | 188.0 | Hombre | 12.2 | 35.8 |
| **1** | 65 | 80.0 | 174.0 | Hombre | 12.0 | 35.0 |
| **2** | 45 | 82.3 | 176.5 | Hombre | 11.2 | 38.5 |
| **3** | 37 | 73.6 | 180.3 | Hombre | 11.2 | 32.2 |
| **4** | 55 | 74.1 | 167.6 | Hombre | 11.8 | 32.9 |

In [12]:
```
# Análisis descriptivo
print("Análisis Descriptivo:")
print(data.describe())

# Seleccionar las columnas numéricas para el análisis
variables = ['edad', 'peso', 'altura', 'muneca', 'biceps']
datos_numeric = data[variables]

# Calcular la desviación estándar
print("\nDesviación Estándar:")
print(datos_numeric.std())
```

```
Análisis Descriptivo:
            edad       peso      altura     muneca     biceps
count  36.000000  36.000000   36.000000  36.000000  36.000000
mean   31.444444  68.952778  171.555556  10.466667  31.166667
std    10.554469  14.868999   10.520170   1.175463   5.234392
min    19.000000  42.000000  147.200000   8.300000  23.500000
25%    24.750000  54.950000  164.800000   9.475000  25.975000
50%    28.000000  71.500000  172.700000  10.650000  32.150000
75%    37.000000  82.400000  179.400000  11.500000  35.050000
max    65.000000  98.200000  190.500000  12.400000  40.400000

Desviación Estándar:
edad      10.554469
peso      14.868999
altura    10.520170
muneca     1.175463
biceps     5.234392
dtype: float64
```

In [4]:
```python
# Matriz de varianza-covarianza
cov_matrix = np.cov(datos_numeric.T)
print("\nMatriz de Varianza-Covarianza:")
print(cov_matrix)

# Matriz de correlaciones
corr_matrix = np.corrcoef(datos_numeric.T)
print("\nMatriz de Correlaciones:")
print(corr_matrix)
```

```
Matriz de Varianza-Covarianza:
[[111.3968254   80.8815873   36.66603175   7.69809524  26.72095238]
 [ 80.8815873  221.08713492 124.72869841  14.84466667  70.73838095]
 [ 36.66603175 124.72869841 110.67396825   8.15647619  39.02104762]
 [  7.69809524  14.84466667   8.15647619   1.38171429   5.40057143]
 [ 26.72095238  70.73838095  39.02104762   5.40057143  27.39885714]]

Matriz de Correlaciones:
[[1.         0.5153847  0.33022106 0.62049423 0.48367017]
 [0.5153847  1.         0.79737371 0.84933611 0.90888127]
 [0.33022106 0.79737371 1.         0.6595849  0.70861438]
 [0.62049423 0.84933611 0.6595849  1.         0.87773692]
 [0.48367017 0.90888127 0.70861438 0.87773692 1.        ]]
```

In [5]:
```python
# Cálculo de valores y vectores propios para la matriz de varianza-covarianza
eigen_values_cov, eigen_vectors_cov = np.linalg.eig(cov_matrix)

# Cálculo de valores y vectores propios para la matriz de correlaciones
eigen_values_corr, eigen_vectors_corr = np.linalg.eig(corr_matrix)

# Mostrar los valores y vectores propios
print("\nValores propios (Varianza-Covarianza):")
print(eigen_values_cov)

print("\nVectores propios (Varianza-Covarianza):")
print(eigen_vectors_cov)

print("\nValores propios (Correlación):")
print(eigen_values_corr)

print("\nVectores propios (Correlación):")
print(eigen_vectors_corr)
```

```
Valores propios (Varianza-Covarianza):
[3.59398024e+02 8.03757858e+01 2.76229011e+01 2.34357051e-01
 4.30743178e+00]

Vectores propios (Varianza-Covarianza):
[[ 0.34871002  0.90755007  0.23248825  0.02647394 -0.00158947]
 [ 0.76617586 -0.16165811 -0.52166894  0.01070786 -0.3385086 ]
 [ 0.47632405 -0.38517546  0.78905759  0.00354315  0.04616081]
 [ 0.05386189  0.0155423  -0.02785902 -0.99003996  0.12610348]
 [ 0.24817367 -0.0402221  -0.22455005  0.13781436  0.9313305 ]]

Valores propios (Correlación):
[3.75749733 0.72585665 0.32032981 0.12461873 0.07169749]

Vectores propios (Correlación):
[[-0.33593103 -0.85756006 -0.3491378  -0.13601109  0.10651229]
 [-0.4927066   0.16478214  0.06924561 -0.52495335 -0.67060874]
 [-0.42224257  0.45422228 -0.73394453  0.20706728  0.18396169]
 [-0.48219233 -0.10827747  0.36690716  0.75515465 -0.22558177]
 [-0.4833139   0.13926843  0.44722747 -0.30461378  0.67395105]]
```

In [6]:
```python
# Proporción de varianza explicada para la matriz de varianza-covarianza
```

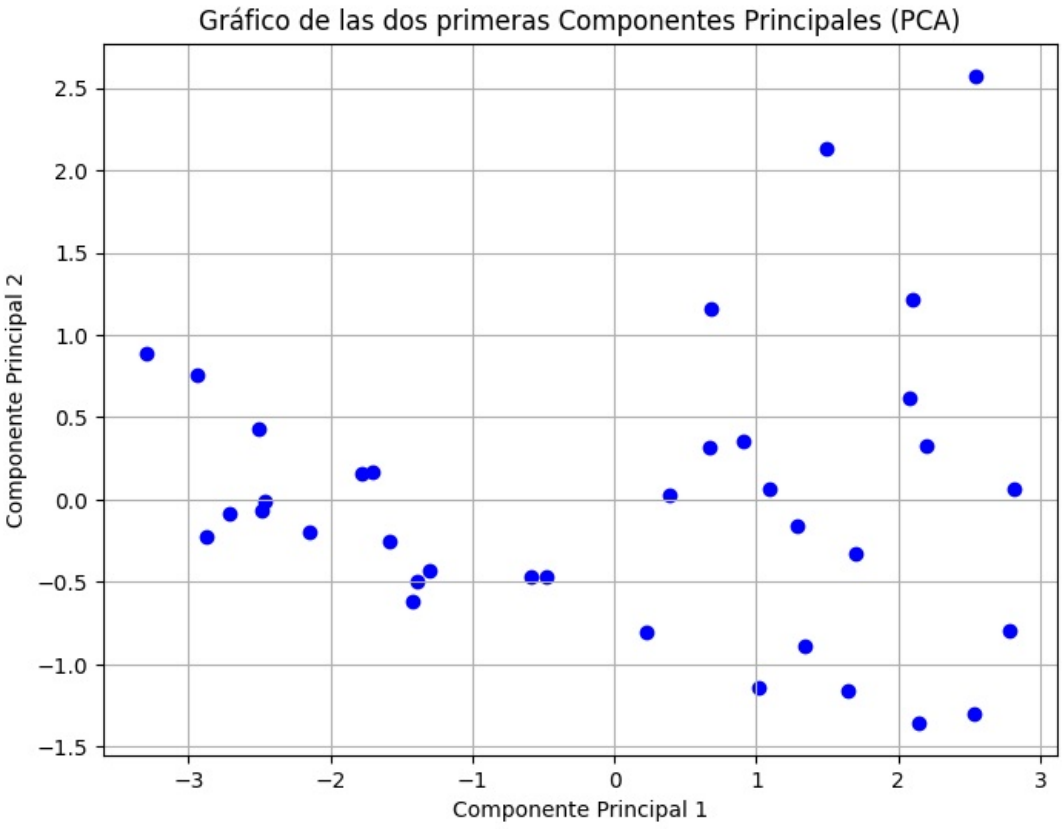
```
var_explained_cov = eigen_values_cov / np.sum(eigen_values_cov)

# Proporción de varianza explicada para la matriz de correlaciones
var_explained_corr = eigen_values_corr / np.sum(eigen_values_corr)

print("\nProporción de varianza explicada (Varianza-Covarianza):")
print(var_explained_cov)

print("\nProporción de varianza explicada (Correlación):")
print(var_explained_corr)
```

```
Proporción de varianza explicada (Varianza-Covarianza):
[7.61535718e-01 1.70309873e-01 5.85307219e-02 4.96583879e-04
 9.12710403e-03]

Proporción de varianza explicada (Correlación):
[0.75149947 0.14517133 0.06406596 0.02492375 0.0143395 ]
```

# Parte II

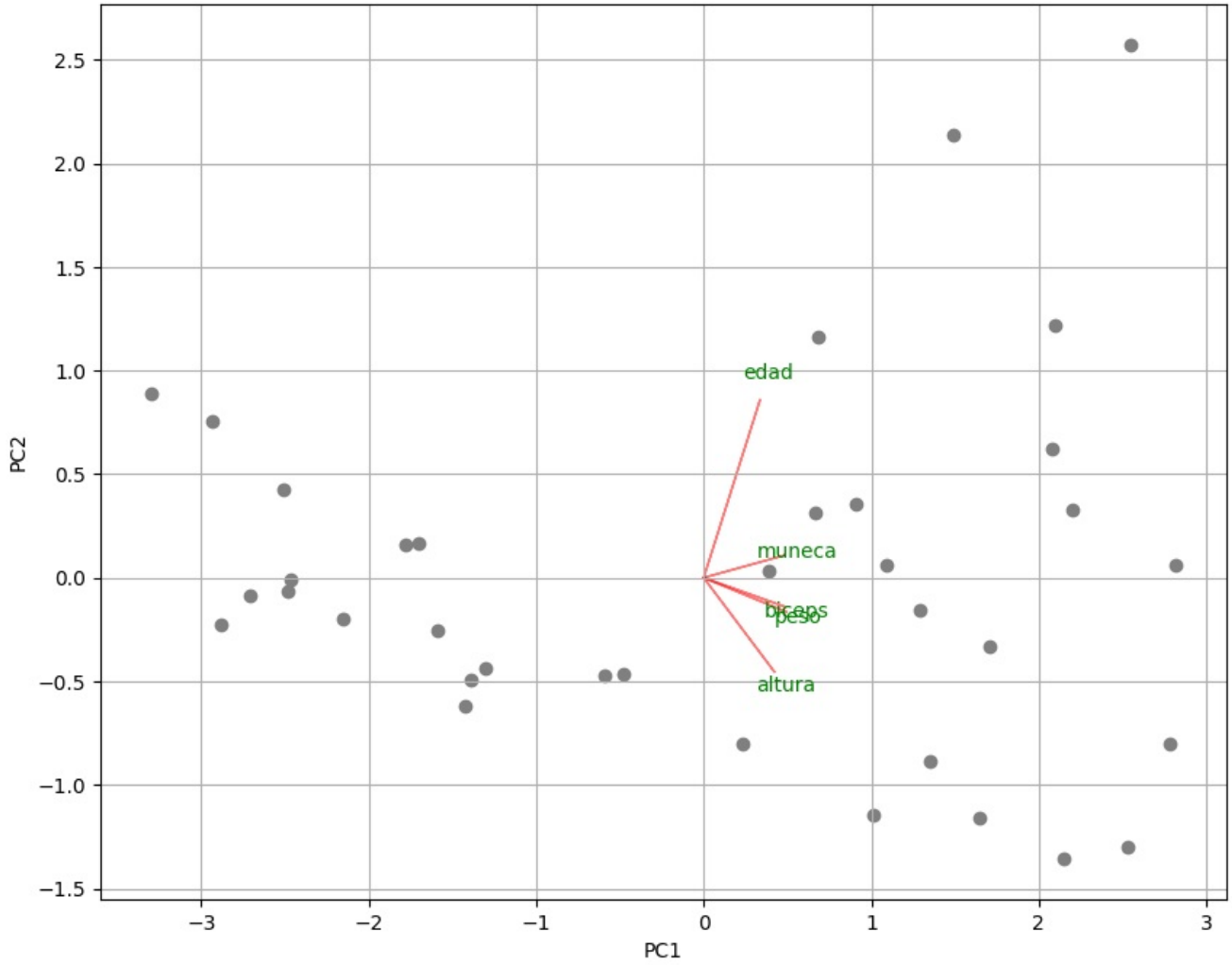
In [7]:
```
# Escalar los datos
scaler = StandardScaler()
datos_scaled = scaler.fit_transform(datos_numeric)

# Aplicar PCA
pca = PCA()
pca.fit(datos_scaled)

# Resultados del PCA
print("\nVarianza explicada por cada componente (PCA con correlación):")
print(pca.explained_variance_ratio_)

# Transformar los datos para obtener las componentes principales
pca_data = pca.transform(datos_scaled)

# Crear un DataFrame con las dos primeras componentes
pca_df = pd.DataFrame(pca_data, columns=[f'PC{i+1}' for i in range(pca_data.shape[1])])

# Graficar las dos primeras componentes principales
plt.figure(figsize=(8, 6))
plt.scatter(pca_df['PC1'], pca_df['PC2'], c='blue')
plt.title('Gráfico de las dos primeras Componentes Principales (PCA)')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.grid(True)
plt.show()
```

```
Varianza explicada por cada componente (PCA con correlación):
[0.75149947 0.14517133 0.06406596 0.02492375 0.0143395 ]
```



# Parte III

In [8]:
```
# Función para generar un Biplot
def biplot(pca_data, eigen_vectors, labels):
    plt.figure(figsize=(10, 8))
    plt.scatter(pca_data[:, 0], pca_data[:, 1], c='gray')

    # Graficar los vectores de las variables originales
    for i in range(len(eigen_vectors)):
        plt.arrow(0, 0, eigen_vectors[i, 0], eigen_vectors[i, 1],
                  color='r', alpha=0.5)
        plt.text(eigen_vectors[i, 0] * 1.15, eigen_vectors[i, 1] * 1.15,
                 labels[i], color='g', ha='center', va='center')

    plt.xlabel('PC1')
    plt.ylabel('PC2')
    plt.grid(True)
    plt.show()

# Generar el Biplot
biplot(pca_data, pca.components_.T, variables)
```




# Paso IV

Matriz de Varianza-Covarianza

Valores Propios: Los valores propios indican la cantidad de varianza explicada por cada componente principal. En este caso, los valores propios son: [359.398024, 80.3757858, 27.6229011, 0.234357051, 4.30743178].

Proporción de Varianza Explicada: La proporción de varianza explicada por cada componente se calcula dividiendo cada valor propio entre la suma total de los valores propios. Proporciones: [0.761535718, 0.170309873, 0.0585307219, 0.000496583879, 0.00912710403].

Componentes Principales Importantes: Los primeros dos componentes principales explican la mayor parte de la varianza (76.15% y 17.03% respectivamente). Estos componentes son los más importantes para interpretar los datos.

Combinaciones Lineales: CP1: 0.3487*edad* + *0.7662*peso + 0.4763*altura* + *0.0539*muñeca + 0.2482*bíceps CP2: 0.9076*edad -

0.1617*peso - 0.3852*altura + 0.0155*muñeca - 0.0402*bíceps Las variables que más contribuyen a CP1 son peso y altura, mientras que para CP2 son edad y peso.

Matriz de Correlación

Valores Propios: Los valores propios son: [3.75749733, 0.72585665, 0.32032981, 0.12461873, 0.07169749].

Proporción de Varianza Explicada: Proporciones: [0.75149947, 0.14517133, 0.06406596, 0.02492375, 0.0143395].

Componentes Principales Importantes: Los primeros dos componentes principales explican la mayor parte de la varianza (75.15% y 14.52% respectivamente).

Combinaciones Lineales:

CP1: -0.3359*edad - 0.4927*peso - 0.4222*altura - 0.4822*muñeca - 0.4833*bíceps CP2: -0.8576*edad + 0.1648*peso + 0.4542*altura - 0.1083*muñeca + 0.1393*bíceps Las variables que más contribuyen a CP1 son bíceps y muñeca, mientras que para CP2 son edad y altura.

Conclusiones Comparación de Procedimientos: Ambos procedimientos (varianza-covarianza y correlación) son útiles, pero la matriz de correlación es más adecuada cuando las variables tienen diferentes unidades o escalas. En este caso, la matriz de correlación parece proporcionar componentes más equilibrados y fáciles de interpretar.

Componentes de Mayor Interés: La matriz de correlación aporta componentes con mayor interés debido a la estandarización de las variables, lo que facilita la comparación.

Variables que Más Contribuyen: Para la matriz de varianza-covarianza, peso y altura son las variables más importantes. Para la matriz de correlación, bíceps y muñeca son las variables más importantes para CP1, y edad y altura para CP2.

Combinaciones Finales: Varianza-Covarianza: CP1: 0.3487*edad + 0.7662*peso + 0.4763*altura + 0.0539*muñeca + 0.2482*bíceps CP2: 0.9076*edad - 0.1617*peso - 0.3852*altura + 0.0155*muñeca - 0.0402*bíceps Correlación: CP1: -0.3359*edad - 0.4927*peso - 0.4222*altura - 0.4822*muñeca - 0.4833*bíceps CP2: -0.8576*edad + 0.1648*peso + 0.4542*altura - 0.1083*muñeca + 0.1393*bíceps

Interpretación de Resultados Agrupación de Variables: Índice de Riqueza: Podría estar relacionado con peso y altura. Índice de Ruralidad: Podría estar relacionado con edad y muñeca.