In [52]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import os
os.chdir('/Users/Lenovo/Desktop/EBAC')
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings('ignore')
```

In [54]:
```
data = pd.read_csv('Iris3.csv')
data.head()
```

Out[54]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

In [56]:
```
data_final = data[['sepal.length', 'sepal.width', 'petal.length', 'petal.width']]
data_final
```

Out[56]:

	sepal.length	sepal.width	petal.length	petal.width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

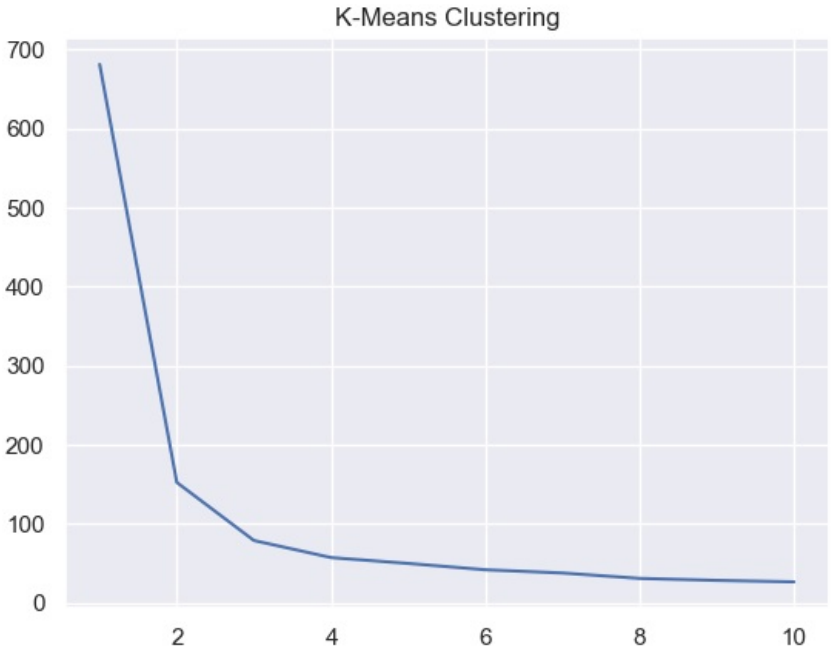
In [58]:
```
wcss = []
for index in range(1, 11):
    kmeans = KMeans(n_clusters = index, init = 'k-means++', random_state = 2)
    kmeans.fit(data_final)

    wcss.append(kmeans.inertia_)
```

In [60]:
```
# Graficacion del "Codo de Jambu"
sb.set()
plt.plot(range(1, 11), wcss)
plt.title('K-Means Clustering')
plt.xlabel('Numero de clusters')
plt.ylabel('WCSS') # Weighted cluster sum of squares
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[60], line 5
      3 plt.plot(range(1, 11), wcss)
      4 plt.title('K-Means Clustering')
----> 5 plt.xlabel('Numero de clusters')
      6 plt.ylabel('WCSS')

TypeError: 'str' object is not callable
```



In [62]:
```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
cluster_values = kmeans.fit_predict(data_final)
print(cluster_values)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 0 0 0 0 2 0 0 0 0
 0 0 2 2 0 0 0 0 2 0 2 0 2 0 0 2 2 0 0 0 0 2 0 0 0 0 2 0 0 0 2 0 0 0 2 0 0 2]
```

In [64]:
```
# Criterio de Silueta ("Silhouette")
from sklearn import datasets
from sklearn.metrics import silhouette_score

# Transformamos la informacion a numpy porque Silhouette necesita la informacion en este formato
X = data_final.to_numpy()
for j in range(2, 12):
    kmeans = KMeans(n_clusters = j, random_state = 42)
    kmeans.fit_predict(X)
    # Calculamos el Score de Silueta
    score = silhouette_score(X, kmeans.labels_, metric = 'euclidean')
    print('Score Silhouette: ', 'k =', j, ':', score)
```

```
Score Silhouette:  k = 2 : 0.6810461692117467
Score Silhouette:  k = 3 : 0.5511916046195927
Score Silhouette:  k = 4 : 0.49764331793219296
Score Silhouette:  k = 5 : 0.493080406719353
Score Silhouette:  k = 6 : 0.3678464984712254
Score Silhouette:  k = 7 : 0.3542978877198859
Score Silhouette:  k = 8 : 0.3446797218056206
Score Silhouette:  k = 9 : 0.3155887853389787
Score Silhouette:  k = 10 : 0.3014143745325155
Score Silhouette:  k = 11 : 0.26873562164120385
```

Tranformacion a dos columnas por PCA

In [66]:
```
fields = data.iloc[:, [0,1,2,3]].values
print(fields)
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
```

```
[4.9 3.1 1.5 0.1]
[5.4 3.7 1.5 0.2]
[4.8 3.4 1.6 0.2]
[4.8 3.  1.4 0.1]
[4.3 3.  1.1 0.1]
[5.8 4.  1.2 0.2]
[5.7 4.4 1.5 0.4]
[5.4 3.9 1.3 0.4]
[5.1 3.5 1.4 0.3]
[5.7 3.8 1.7 0.3]
[5.1 3.8 1.5 0.3]
[5.4 3.4 1.7 0.2]
[5.1 3.7 1.5 0.4]
[4.6 3.6 1.  0.2]
[5.1 3.3 1.7 0.5]
[4.8 3.4 1.9 0.2]
[5.  3.  1.6 0.2]
[5.  3.4 1.6 0.4]
[5.2 3.5 1.5 0.2]
[5.2 3.4 1.4 0.2]
[4.7 3.2 1.6 0.2]
[4.8 3.1 1.6 0.2]
[5.4 3.4 1.5 0.4]
[5.2 4.1 1.5 0.1]
[5.5 4.2 1.4 0.2]
[4.9 3.1 1.5 0.2]
[5.  3.2 1.2 0.2]
[5.5 3.5 1.3 0.2]
[4.9 3.6 1.4 0.1]
[4.4 3.  1.3 0.2]
[5.1 3.4 1.5 0.2]
[5.  3.5 1.3 0.3]
[4.5 2.3 1.3 0.3]
[4.4 3.2 1.3 0.2]
[5.  3.5 1.6 0.6]
[5.1 3.8 1.9 0.4]
[4.8 3.  1.4 0.3]
[5.1 3.8 1.6 0.2]
[4.6 3.2 1.4 0.2]
[5.3 3.7 1.5 0.2]
[5.  3.3 1.4 0.2]
[7.  3.2 4.7 1.4]
[6.4 3.2 4.5 1.5]
[6.9 3.1 4.9 1.5]
[5.5 2.3 4.  1.3]
[6.5 2.8 4.6 1.5]
[5.7 2.8 4.5 1.3]
[6.3 3.3 4.7 1.6]
[4.9 2.4 3.3 1. ]
[6.6 2.9 4.6 1.3]
[5.2 2.7 3.9 1.4]
[5.  2.  3.5 1. ]
[5.9 3.  4.2 1.5]
[6.  2.2 4.  1. ]
[6.1 2.9 4.7 1.4]
[5.6 2.9 3.6 1.3]
[6.7 3.1 4.4 1.4]
[5.6 3.  4.5 1.5]
[5.8 2.7 4.1 1. ]
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.  1.3]
[6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.  4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.  5.  1.7]
[6.  2.9 4.5 1.5]
[5.7 2.6 3.5 1. ]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1. ]
[5.8 2.7 3.9 1.2]
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
[6.  3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.  4.1 1.3]
[5.5 2.5 4.  1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.  4.6 1.4]
```

```
[5.8 2.6 4.  1.2]
[5.  2.3 3.3 1. ]
[5.6 2.7 4.2 1.3]
[5.7 3.  4.2 1.2]
[5.7 2.9 4.2 1.3]
[6.2 2.9 4.3 1.3]
[5.1 2.5 3.  1.1]
[5.7 2.8 4.1 1.3]
[6.3 3.3 6.  2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.  5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.  5.8 2.2]
[7.6 3.  6.6 2.1]
[4.9 2.5 4.5 1.7]
[7.3 2.9 6.3 1.8]
[6.7 2.5 5.8 1.8]
[7.2 3.6 6.1 2.5]
[6.5 3.2 5.1 2. ]
[6.4 2.7 5.3 1.9]
[6.8 3.  5.5 2.1]
[5.7 2.5 5.  2. ]
[5.8 2.8 5.1 2.4]
[6.4 3.2 5.3 2.3]
[6.5 3.  5.5 1.8]
[7.7 3.8 6.7 2.2]
[7.7 2.6 6.9 2.3]
[6.  2.2 5.  1.5]
[6.9 3.2 5.7 2.3]
[5.6 2.8 4.9 2. ]
[7.7 2.8 6.7 2. ]
[6.3 2.7 4.9 1.8]
[6.7 3.3 5.7 2.1]
[7.2 3.2 6.  1.8]
[6.2 2.8 4.8 1.8]
[6.1 3.  4.9 1.8]
[6.4 2.8 5.6 2.1]
[7.2 3.  5.8 1.6]
[7.4 2.8 6.1 1.9]
[7.9 3.8 6.4 2. ]
[6.4 2.8 5.6 2.2]
[6.3 2.8 5.1 1.5]
[6.1 2.6 5.6 1.4]
[7.7 3.  6.1 2.3]
[6.3 3.4 5.6 2.4]
[6.4 3.1 5.5 1.8]
[6.  3.  4.8 1.8]
[6.9 3.1 5.4 2.1]
[6.7 3.1 5.6 2.4]
[6.9 3.1 5.1 2.3]
[5.8 2.7 5.1 1.9]
[6.8 3.2 5.9 2.3]
[6.7 3.3 5.7 2.5]
[6.7 3.  5.2 2.3]
[6.3 2.5 5.  1.9]
[6.5 3.  5.2 2. ]
[6.2 3.4 5.4 2.3]
[5.9 3.  5.1 1.8]]
```

In [68]:

```
# Algoritmo pCA
from sklearn import decomposition

pca = decomposition.PCA(n_components = 2)
pca.fit(fields)
fields = pca.transform(fields)
fields
```

Out[68]:

```
array([[-2.68412563,  0.31939725],
       [-2.71414169, -0.17700123],
       [-2.88899057, -0.14494943],
       [-2.74534286, -0.31829898],
       [-2.72871654,  0.32675451],
       [-2.28085963,  0.74133045],
       [-2.82053775, -0.08946138],
       [-2.62614497,  0.16338496],
       [-2.88638273, -0.57831175],
       [-2.6727558 , -0.11377425],
       [-2.50694709,  0.6450689 ],
       [-2.61275523,  0.01472994],
       [-2.78610927, -0.235112  ],
       [-3.22380374, -0.51139459],
       [-2.64475039,  1.17876464],
       [-2.38603903,  1.33806233],
```

```
[-2.62352788,  0.81067951],
[-2.64829671,  0.31184914],
[-2.19982032,  0.87283904],
[-2.5879864 ,  0.51356031],
[-2.31025622,  0.39134594],
[-2.54370523,  0.43299606],
[-3.21593942,  0.13346807],
[-2.30273318,  0.09870885],
[-2.35575405, -0.03728186],
[-2.50666891, -0.14601688],
[-2.46882007,  0.13095149],
[-2.56231991,  0.36771886],
[-2.63953472,  0.31203998],
[-2.63198939, -0.19696122],
[-2.58739848, -0.20431849],
[-2.4099325 ,  0.41092426],
[-2.64886233,  0.81336382],
[-2.59873675,  1.09314576],
[-2.63692688, -0.12132235],
[-2.86624165,  0.06936447],
[-2.62523805,  0.59937002],
[-2.80068412,  0.26864374],
[-2.98050204, -0.48795834],
[-2.59000631,  0.22904384],
[-2.77010243,  0.26352753],
[-2.84936871, -0.94096057],
[-2.99740655, -0.34192606],
[-2.40561449,  0.18887143],
[-2.20948924,  0.43666314],
[-2.71445143, -0.2502082 ],
[-2.53814826,  0.50377114],
[-2.83946217, -0.22794557],
[-2.54308575,  0.57941002],
[-2.70335978,  0.10770608],
[ 1.28482569,  0.68516047],
[ 0.93248853,  0.31833364],
[ 1.46430232,  0.50426282],
[ 0.18331772, -0.82795901],
[ 1.08810326,  0.07459068],
[ 0.64166908, -0.41824687],
[ 1.09506066,  0.28346827],
[-0.74912267, -1.00489096],
[ 1.04413183,  0.2283619 ],
[-0.0087454 , -0.72308191],
[-0.50784088, -1.26597119],
[ 0.51169856, -0.10398124],
[ 0.26497651, -0.55003646],
[ 0.98493451, -0.12481785],
[-0.17392537, -0.25485421],
[ 0.92786078,  0.46717949],
[ 0.66028376, -0.35296967],
[ 0.23610499, -0.33361077],
[ 0.94473373, -0.54314555],
[ 0.04522698, -0.58383438],
[ 1.11628318, -0.08461685],
[ 0.35788842, -0.06892503],
[ 1.29818388, -0.32778731],
[ 0.92172892, -0.18273779],
[ 0.71485333,  0.14905594],
[ 0.90017437,  0.32850447],
[ 1.33202444,  0.24444088],
[ 1.55780216,  0.26749545],
[ 0.81329065, -0.1633503 ],
[-0.30558378, -0.36826219],
[-0.06812649, -0.70517213],
[-0.18962247, -0.68028676],
[ 0.13642871, -0.31403244],
[ 1.38002644, -0.42095429],
[ 0.58800644, -0.48428742],
[ 0.80685831,  0.19418231],
[ 1.22069088,  0.40761959],
[ 0.81509524, -0.37203706],
[ 0.24595768, -0.2685244 ],
[ 0.16641322, -0.68192672],
[ 0.46480029, -0.67071154],
[ 0.8908152 , -0.03446444],
[ 0.23054802, -0.40438585],
[-0.70453176, -1.01224823],
[ 0.35698149, -0.50491009],
[ 0.33193448, -0.21265468],
[ 0.37621565, -0.29321893],
[ 0.64257601,  0.01773819],
[-0.90646986, -0.75609337],
```

```
        [ 0.29900084, -0.34889781],
        [ 2.53119273, -0.00984911],
        [ 1.41523588, -0.57491635],
        [ 2.61667602,  0.34390315],
        [ 1.97153105, -0.1797279 ],
        [ 2.35000592, -0.04026095],
        [ 3.39703874,  0.55083667],
        [ 0.52123224, -1.19275873],
        [ 2.93258707,  0.3555    ],
        [ 2.32122882, -0.2438315 ],
        [ 2.91675097,  0.78279195],
        [ 1.66177415,  0.24222841],
        [ 1.80340195, -0.21563762],
        [ 2.1655918 ,  0.21627559],
        [ 1.34616358, -0.77681835],
        [ 1.58592822, -0.53964071],
        [ 1.90445637,  0.11925069],
        [ 1.94968906,  0.04194326],
        [ 3.48705536,  1.17573933],
        [ 3.79564542,  0.25732297],
        [ 1.30079171, -0.76114964],
        [ 2.42781791,  0.37819601],
        [ 1.19900111, -0.60609153],
        [ 3.49992004,  0.4606741 ],
        [ 1.38876613, -0.20439933],
        [ 2.2754305 ,  0.33499061],
        [ 2.61409047,  0.56090136],
        [ 1.25850816, -0.17970479],
        [ 1.29113206, -0.11666865],
        [ 2.12360872, -0.20972948],
        [ 2.38800302,  0.4646398 ],
        [ 2.84167278,  0.37526917],
        [ 3.23067366,  1.37416509],
        [ 2.15943764, -0.21727758],
        [ 1.44416124, -0.14341341],
        [ 1.78129481, -0.49990168],
        [ 3.07649993,  0.68808568],
        [ 2.14424331,  0.1400642 ],
        [ 1.90509815,  0.04930053],
        [ 1.16932634, -0.16499026],
        [ 2.10761114,  0.37228787],
        [ 2.31415471,  0.18365128],
        [ 1.9222678 ,  0.40920347],
        [ 1.41523588, -0.57491635],
        [ 2.56301338,  0.2778626 ],
        [ 2.41874618,  0.3047982 ],
        [ 1.94410979,  0.1875323 ],
        [ 1.52716661, -0.37531698],
        [ 1.76434572,  0.07885885],
        [ 1.90094161,  0.11662796],
        [ 1.39018886, -0.28266094]])
```

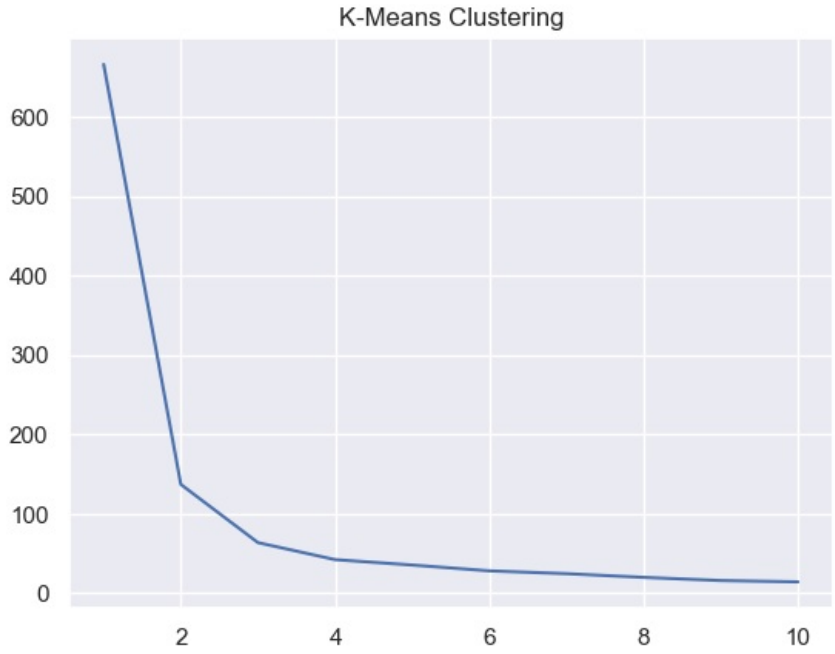
In [70]:
```
# Calculo de las distancias de cada punto a cada centroide
wcss2 = []
for index in range(1, 11):
    kmeans = KMeans(n_clusters = index, init = 'k-means++', random_state = 2)
    kmeans.fit(fields)

    wcss2.append(kmeans.inertia_)
```

In [72]:
```
# Graficacion del "Codo de Jambu"
sb.set()
plt.plot(range(1, 11), wcss2)
plt.title('K-Means Clustering')
plt.xlabel('Numero de clusters')
plt.ylabel('WCSS2') # Weighted cluster sum of squares
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[72], line 5
      3 plt.plot(range(1, 11), wcss2)
      4 plt.title('K-Means Clustering')
----> 5 plt.xlabel('Numero de clusters')
      6 plt.ylabel('WCSS2')

TypeError: 'str' object is not callable
```



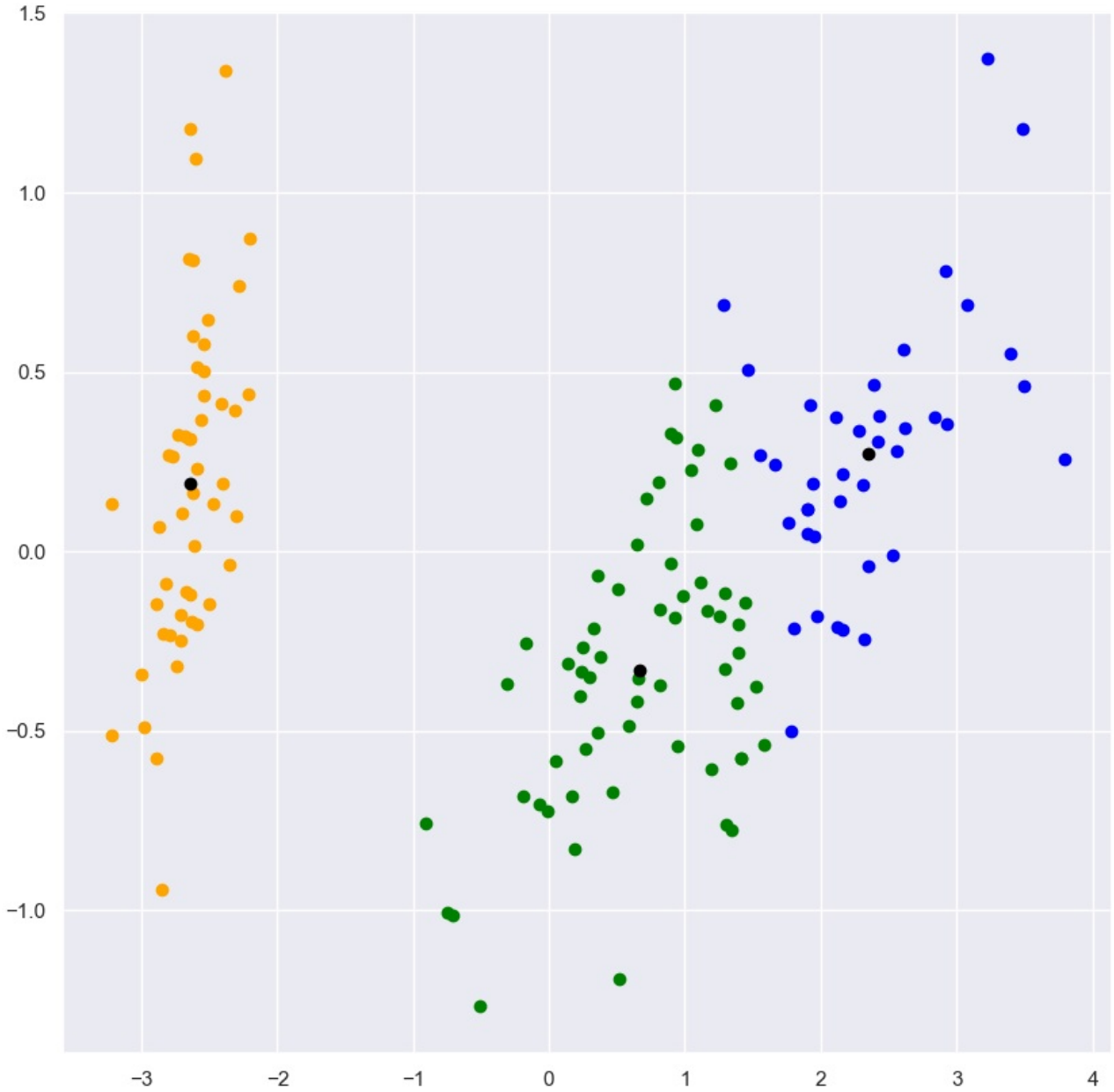
In [74]:
```
#Prediccion de cada grupo
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
cluster_values2 = kmeans.fit_predict(fields)
print(cluster_values2)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 0 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 0 0 0 0 2 0 0 0 0
 0 0 2 2 0 0 0 0 2 0 2 0 2 0 0 2 2 0 0 0 0 0 2 0 0 0 0 2 0 0 0 2 0 0 0 2 0
 0 2]
```

In [76]:
```
plt.figure(figsize = (10,10))
plt.scatter(fields[cluster_values2 == 0,0], fields[cluster_values2 == 0, 1], c = 'blue', label = 'Cluster1')
plt.scatter(fields[cluster_values2 == 1,0], fields[cluster_values2 == 1, 1], c = 'orange', label = 'Cluster2')
plt.scatter(fields[cluster_values2 == 2,0], fields[cluster_values2 == 2, 1], c = 'green', label = 'Cluster3')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c = 'black', label = 'Centroides')
```

Out[76]:
```
<matplotlib.collections.PathCollection at 0x1e871e80980>
```



In [80]:
```
# Transformamos la informacion a numpy porque Silhouette necesita la informacion en este formato
X = fields
for j in range(2, 12):
    kmeans = KMeans(n_clusters = j, random_state = 42)
```

```
    kmeans.fit_predict(X)
    # Calculamos el Score de Silueta
    score = silhouette_score(X, kmeans.labels_, metric = 'euclidean')
    print('Score Silhouette: ', 'k =', j, ':', score)
```

```
Score Silhouette:  k = 2 : 0.7056703225102108
Score Silhouette:  k = 3 : 0.597676421954799
Score Silhouette:  k = 4 : 0.5577409232246783
Score Silhouette:  k = 5 : 0.5100407194899202
Score Silhouette:  k = 6 : 0.40319738737442673
Score Silhouette:  k = 7 : 0.3868263997713608
Score Silhouette:  k = 8 : 0.4407603324965402
Score Silhouette:  k = 9 : 0.40201438763663605
Score Silhouette:  k = 10 : 0.4158865801753246
Score Silhouette:  k = 11 : 0.3994325519305235
```

Conclusion

Aunque los resultados puedan ser un poco similares, el reduciar los datos por PCA es escencial, visualmente (poder hacer una grafica de dispersion en 2 ejes) asi como tambien en estandarizacion de los datos, ya que cada variable tiene un rango distinto que puede llegar a cesgar el resultado, una vez normalizado podemos trabajar con dos variables y datos en una misma escala, que nos permitira un mejor agrupamiento y graficacion.