In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
from sklearn.cluster import KMeans
```

In [4]:
```
dataset = pd.read_csv('C:/Users/Isaac/Desktop/IHD/EBAC DT/CIENCIA DE DATOS/M27 DS/iris.csv')
dataset.head()
```

Out[4]:

	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 [9]:
```
fields = dataset.iloc[:,[2,3]].values
fields
```

Out[9]:
```
array([[1.4, 0.2],
       [1.4, 0.2],
       [1.3, 0.2],
       [1.5, 0.2],
       [1.4, 0.2],
       [1.7, 0.4],
       [1.4, 0.3],
       [1.5, 0.2],
       [1.4, 0.2],
       [1.5, 0.1],
       [1.5, 0.2],
       [1.6, 0.2],
       [1.4, 0.1],
       [1.1, 0.1],
```

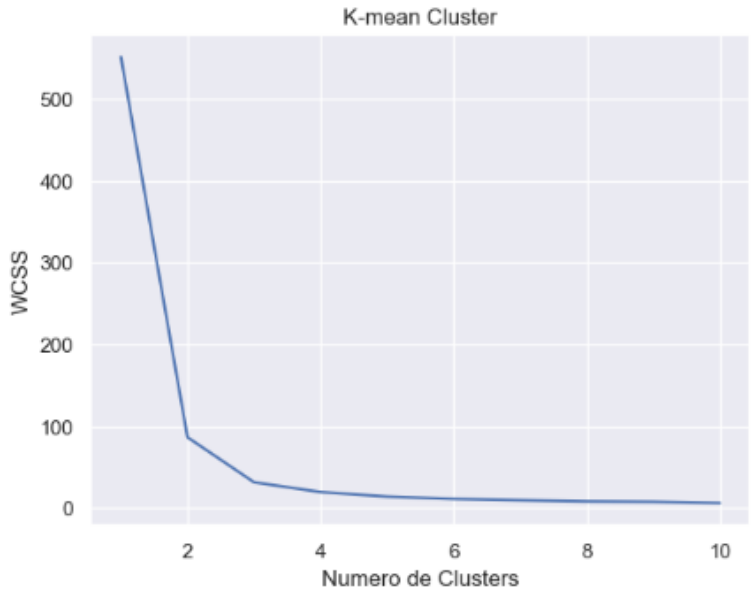
In [10]:
```
wcss = []

for i in range(1,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 2)
    kmeans.fit(fields)
    wcss.append(kmeans.inertia_)
```

- GRAFICO DE CODO

In [12]:
```
sns.set()
plt.plot(range(1,11), wcss)
plt.title('K-mean Cluster')
plt.xlabel('Numero de Clusters')
plt.ylabel('WCSS')
```

Out[12]: Text(0, 0.5, 'WCSS')



El número de cluster optimo según la grafica de CODO es de 3

In [15]:
```
# clasificación de grupos

kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
cluster_values = kmeans.fit_predict(fields)
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 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
 2 2 2 0 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 2 0 0 0 0
 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0
 0 0]
```

0 0]

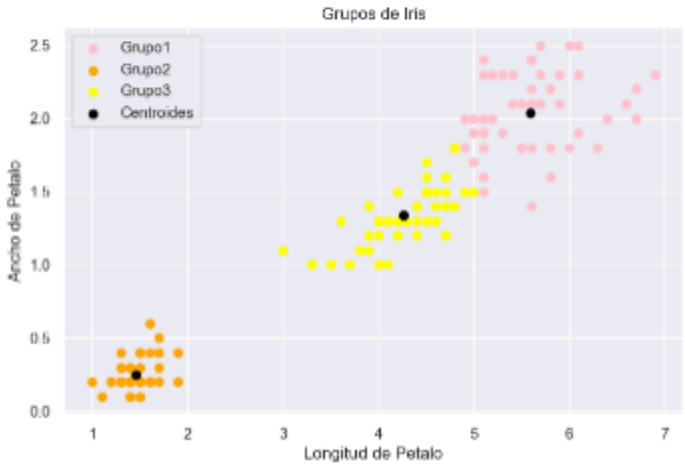
In [17]:
```
# grtoficamos decuardo a los grupos generados

plt.figure(figsize = (8,5))
plt.scatter(fields[cluster_values == 0, 0], fields[cluster_values == 0, 1], c = 'pink', label = 'Grupo1')
plt.scatter(fields[cluster_values == 1, 0], fields[cluster_values == 1, 1], c = 'orange', label = 'Grupo2')
plt.scatter(fields[cluster_values == 2, 0], fields[cluster_values == 2, 1], c = 'yellow', label = 'Grupo3')

# visualizamos centriodes

plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], c = 'black', label = 'Centroides')
plt.title('Grupos de Iris')
plt.xlabel('Longitud de Petalo')
plt.ylabel('Ancho de Petalo')
plt.legend()
```

Out[17]: <matplotlib.legend.Legend at 0x17fa2a983d8>



- Utilzamos el criterio de Silueta

In [19]:
```
from sklearn import datasets
from sklearn.metrics import silhouette_score
```

In [21]:
```
X = dataset.iloc[:,[2, 3]].to_numpy()
for j in range(2,12):
    kmeans = KMeans(n_clusters = j, random_state = 42)
    kmeans.fit_predict(X)
    score = silhouette_score(X, kmeans.labels_, metric = 'euclidean')
    print('Score Silhouette:', 'k =', j, ':', score)
```

```
Score Silhouette: k = 2 : 0.7653904101383076
Score Silhouette: k = 3 : 0.6604800085022658
Score Silhouette: k = 4 : 0.6127580795614039
Score Silhouette: k = 5 : 0.588373271407563
Score Silhouette: k = 6 : 0.5617282187055354
Score Silhouette: k = 7 : 0.5549940910739419
Score Silhouette: k = 8 : 0.5245973296356812
Score Silhouette: k = 9 : 0.5297498184921089
Score Silhouette: k = 10 : 0.5174205026033651
Score Silhouette: k = 11 : 0.5134138267091934
```

- Este valor 'Score Silhouette: k = 3 : 0.6604800085022658' es el mas cercano al 1, lo que indica una mejor agrupación, el resultado indica que son 3 clusters el valor optimo lo que confirma con el mismo resultado del algoritmo kmeans(3 clusters valor optimo)

In [22]:
```
# algoritmo KMeans aplicado a los valores originales de la BD riginal, bajo la transformación de PCA

fields2 = dataset.iloc[:, [0,1,2,3]].values
fields2
```

Out[22]:
```
array([[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],
```

In [23]:
```
# plicamos PCA

from sklearn import decomposition
pca = decomposition.PCA(n_components = 2)
pca.fit(fields2)
fields2 = pca.transform(fields2)
fields2
```

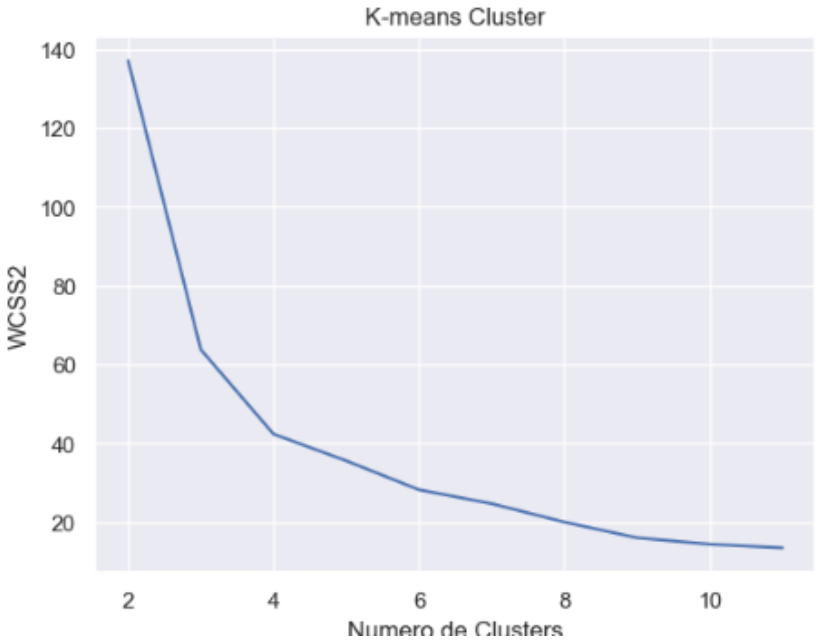
Out[23]:
```
array([[-2.68412563,  0.31939725],
       [-2.71414169, -0.17700123],
       [-2.88899057, -0.14494943],
       [-2.74534286, -0.31829898],
       [-2.72871654,  0.32675451],
       [-2.28085963,  0.74133045],
       [-2.82053775, -0.08946138],
       [-2.62614497,  0.16338496],
       [-2.88638273, -0.57831175],
       [-2.6727558 , -0.11377425],
       [-2.50694709,  0.6450689 ],
       [-2.61275523,  0.01472994],
```

In [25]:
```
wcss2 = []
for index in range(2,12):
    kmeans = KMeans(n_clusters = index, init = 'k-means++', random_state = 2)
    kmeans.fit(fields2)
    wcss2.append(kmeans.inertia_)
```

In [27]:
```
# grafico de CODO

sns.set()
plt.plot(range(2,12), wcss2)
plt.title('K-means Cluster')
plt.xlabel('Numero de Clusters')
plt.ylabel('WCSS2')
```

Out[27]:
```
Text(0, 0.5, 'WCSS2')
```



- Numero de clusters optimo = 3

In [29]:
```
# clasificación por grupos

kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 2)
cluster_values = kmeans.fit_predict(fields2)
cluster_values
```

Out[29]:
```
array([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, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2,
       2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2,
       2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0])
```

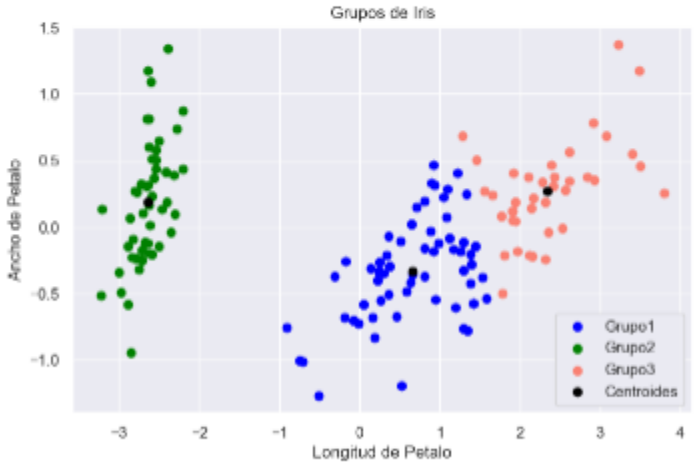
In [33]:
```
# graficamos de cauerdo a los grupos

plt.figure(figsize= (8,5))
plt.scatter(fields2[cluster_values == 0, 0], fields2[cluster_values == 0, 1], c = 'blue', label = 'Grupo1')
plt.scatter(fields2[cluster_values == 1, 0], fields2[cluster_values == 1, 1], c = 'green', label = 'Grupo2')
plt.scatter(fields2[cluster_values == 2, 0], fields2[cluster_values == 2, 1], c = 'salmon', label = 'Grupo3')

# visualizamos los centroides

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], c = 'black', label = 'Centroides')
plt.title('Grupos de Iris')
plt.xlabel('Longitud de Petalo')
plt.ylabel('Ancho de Petalo')
plt.legend()
```

Out[33]:
```
<matplotlib.legend.Legend at 0x17fa489d110>
```



En conclución

- Se obtuvieron los mismos resultados en este ejercicio, en cuanto usar el algoritmo KMeans sin transformar y utilizando PCA