In [7]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import warnings
```

In [3]:
```python
df = pd.read_csv("ML Datasets/iris (2).csv")
```

In [4]:
```python
X = df.iloc[:, 1:-1]
```

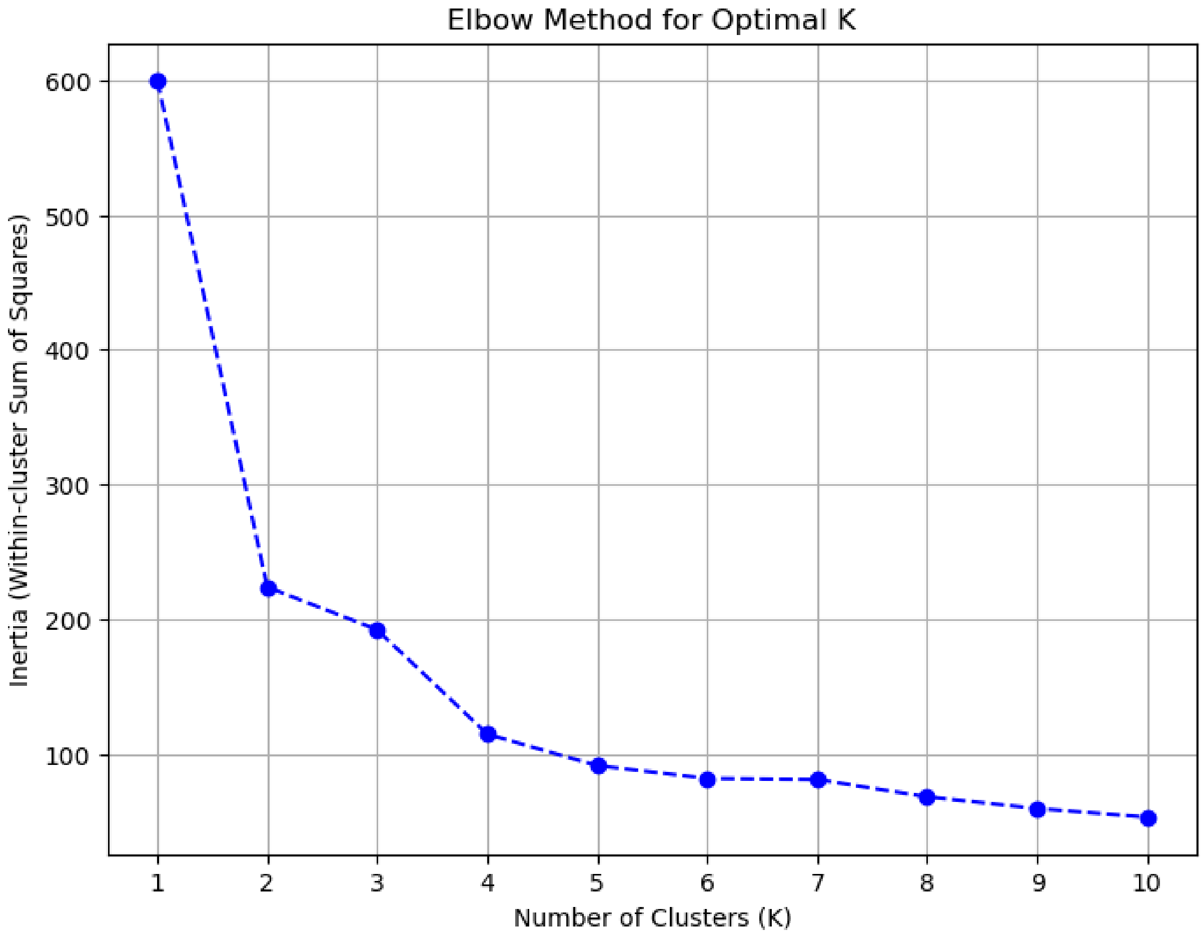
In [5]:
```python
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [8]:
```python
warnings.filterwarnings("ignore")
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)
```

In [9]:
```python
print(inertia)
```

```
[600.0, 223.73200573676345, 192.0371740919003, 114.68221609937964, 91.295444
74066984, 81.7602613286062, 80.98238131032987, 68.08623905064636, 59.3852888
2045365, 52.98999721015859]
```

In [11]:
```python
k_values = range(1, 11)

plt.figure(figsize=(8, 6))
plt.plot(k_values, inertia, marker='o', linestyle='--', color='blue')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia (Within-cluster Sum of Squares)')
plt.xticks(k_values)
plt.grid(True)
plt.show()
```



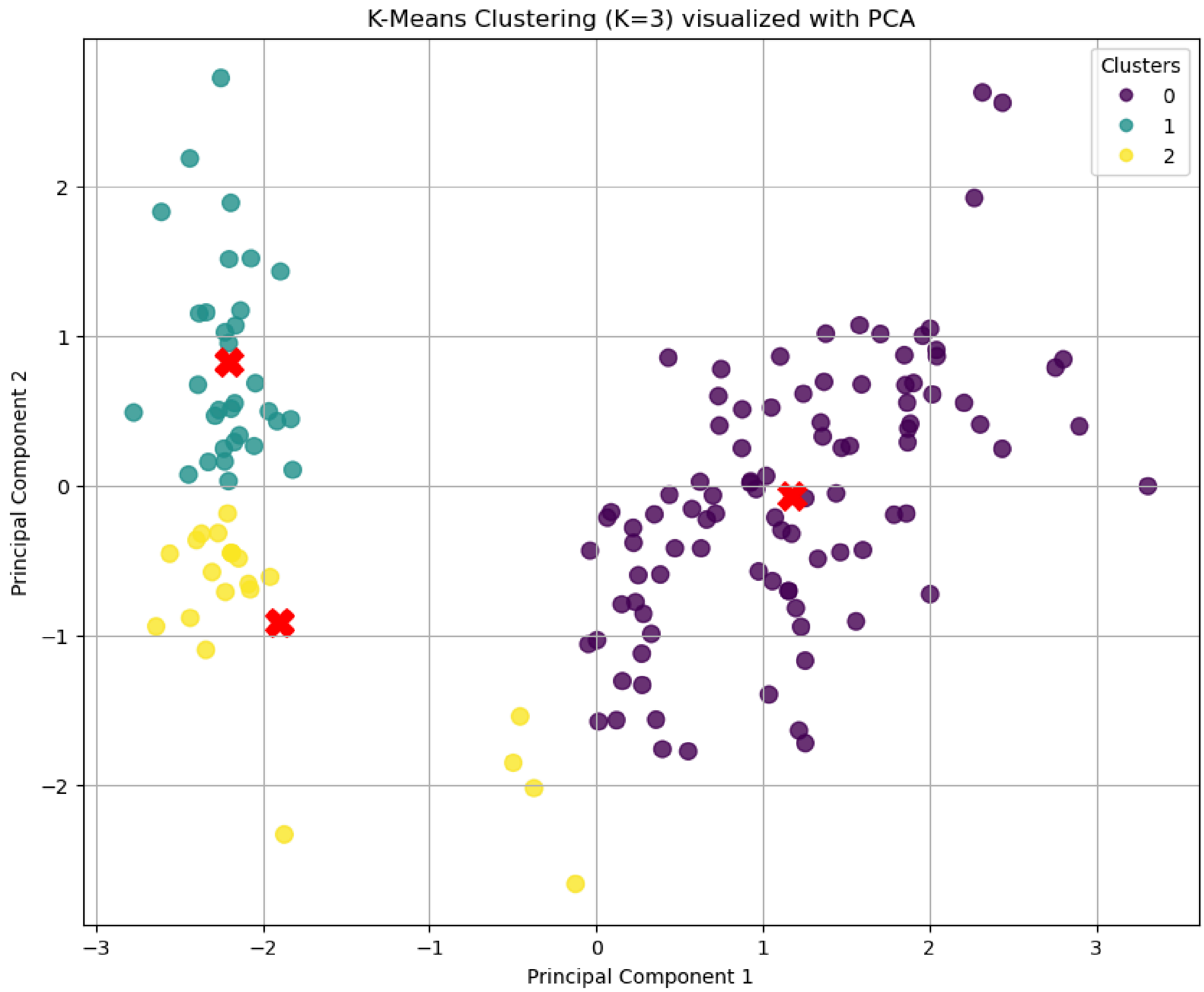
In [12]:
```
optimal_k = 3

kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init='auto')
kmeans.fit(X_scaled)

clusters = kmeans.labels_
df['KMeans_Cluster'] = clusters

print("DataFrame with Cluster Labels:")
print(df.head())
```

```
DataFrame with Cluster Labels:
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species  \
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa

   KMeans_Cluster
0               1
1               2
2               2
3               2
4               1
```

In [13]:
```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
pca_df = pd.DataFrame(data=X_pca, columns=['Principal Component 1', 'Princip

pca_df['KMeans_Cluster'] = clusters
print("\nVariance Explained by Principal Components:")
print(pca.explained_variance_ratio_)
print(f"Total Variance Explained: {pca.explained_variance_ratio_.sum():.2f}"
```

```
Variance Explained by Principal Components:
[0.72770452 0.23030523]
Total Variance Explained: 0.96
```

In [14]:
```
plt.figure(figsize=(10, 8))
scatter = plt.scatter(
    pca_df['Principal Component 1'],
    pca_df['Principal Component 2'],
    c=pca_df['KMeans_Cluster'],
    cmap='viridis',
    s=70,
    alpha=0.8
)

centers_pca = pca.transform(kmeans.cluster_centers_)
plt.scatter(
    centers_pca[:, 0],
    centers_pca[:, 1],
    marker='X',
    s=200,
    c='red',
    label='Cluster Centers'
)

plt.title('K-Means Clustering (K=3) visualized with PCA')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(*scatter.legend_elements(), title="Clusters")
plt.grid(True)
plt.show()
```


K-Means Clustering (K=3) visualized with PCA
Clusters
0
1
2
Principal Component 1
Principal Component 2
−3
−2
−1
0
1
2
3