In [67]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
```

In [69]:
```
from sklearn.datasets import load_wine

data = load_wine()
```

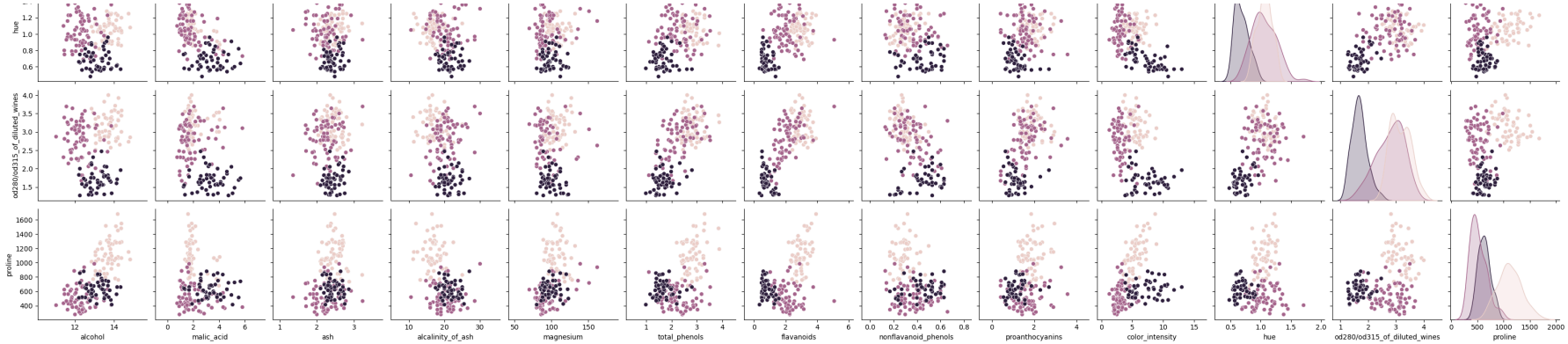
In [11]:
```
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
df.head()
```

Out[11]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensi
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.6
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.3
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.6
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.8
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.3

In [19]:
```
# Visualizar os dados
sns.pairplot(df, hue='target')
plt.show()
```


alcohol
malic_acid
ash
alcalinity_of_ash
magnesium
total_phenols
flavanoids
nonflavanoid_phenols
proanthocyanins
color_intensity
target
0
1
2



In [21]:
```
# Normalizar os dados
scaler = StandardScaler()
df[data.feature_names] = scaler.fit_transform(df[data.feature_names])
df[data.feature_names].head()
```

Out[21]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_i
0	1.518613	-0.562250	0.232053	-1.169593	1.913905	0.808997	1.034819	-0.659563	1.224884	
1	0.246290	-0.499413	-0.827996	-2.490847	0.018145	0.568648	0.733629	-0.820719	-0.544721	-
2	0.196879	0.021231	1.109334	-0.268738	0.088358	0.808997	1.215533	-0.498407	2.135968	
3	1.691550	-0.346811	0.487926	-0.809251	0.930918	2.491446	1.466525	-0.981875	1.032155	
4	0.295700	0.227694	1.840403	0.451946	1.281985	0.808997	0.663351	0.226796	0.401404	-

In [35]:
```
X = df[data.feature_names]
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

In [37]:
```
model = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)
model.fit(X_train, y_train)
```

Out[37]:

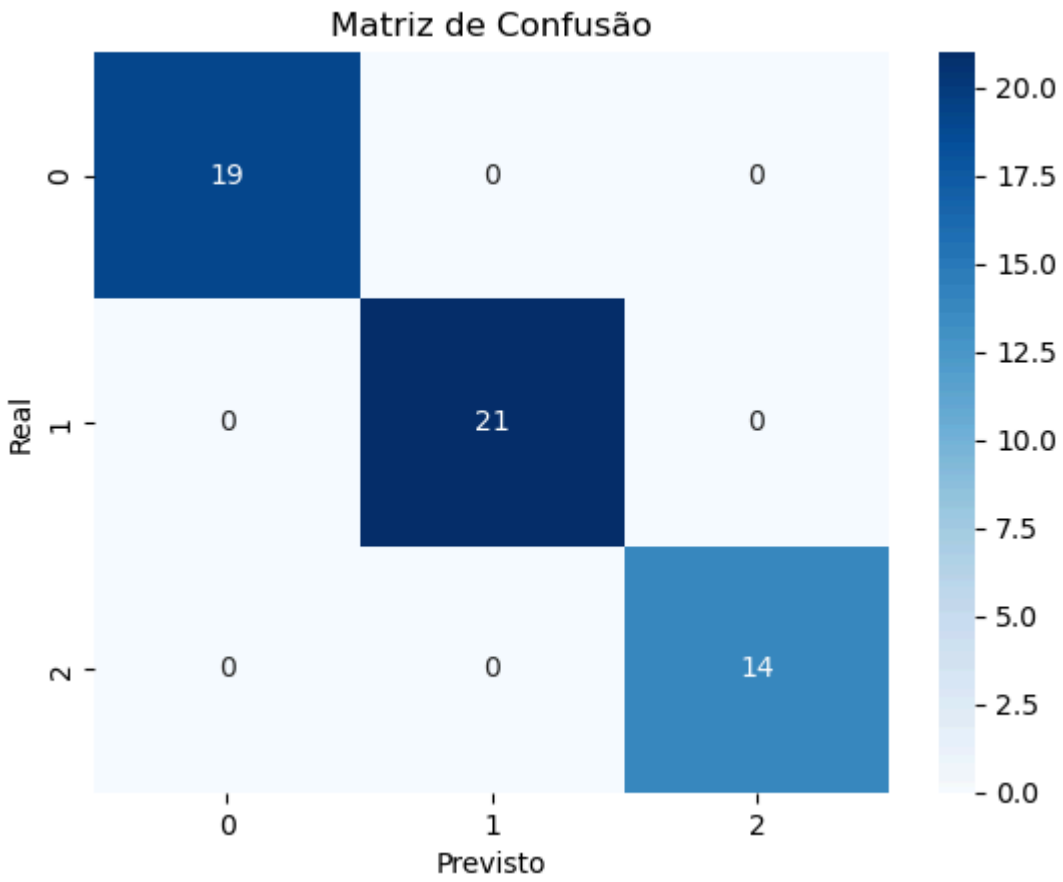
RandomForestClassifier

```
RandomForestClassifier(max_depth=10, random_state=42)
```

In [43]:

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Acurácia: {accuracy * 100:.2f}%')
# Matriz de confusão
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d')
plt.xlabel('Previsto')
plt.ylabel('Real')
plt.title('Matriz de Confusão')
plt.show()
```

```
Acurácia: 100.00%
```



In [47]:
```
from sklearn.model_selection import GridSearchCV
param_grid = {
'n_estimators': [50, 100, 200],
'max_depth': [None, 10, 20, 30]
}
grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid,
cv=5)
grid_search.fit(X_train, y_train)
print(f'Melhores parâmetros: {grid_search.best_params_}')
```

```
Melhores parâmetros: {'max_depth': None, 'n_estimators': 50}
```

In [49]:
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X, y, cv=5)
print(f'Acurácia média na validação cruzada: {scores.mean() * 100:.2f}%')
```

```
Acurácia média na validação cruzada: 97.21%
```

In [51]:
```
 from sklearn.datasets import fetch_california_housing
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

In [53]:
```
 housing_data = fetch_california_housing()
X = pd.DataFrame(housing_data.data, columns=housing_data.feature_names)
y = pd.Series(housing_data.target)
# Visualizando as primeiras linhas do dataset
print(X.head())
```

```
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85

   Longitude
0    -122.23
1    -122.22
2    -122.24
3    -122.25
4    -122.25
```

In [55]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

In [57]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[57]:

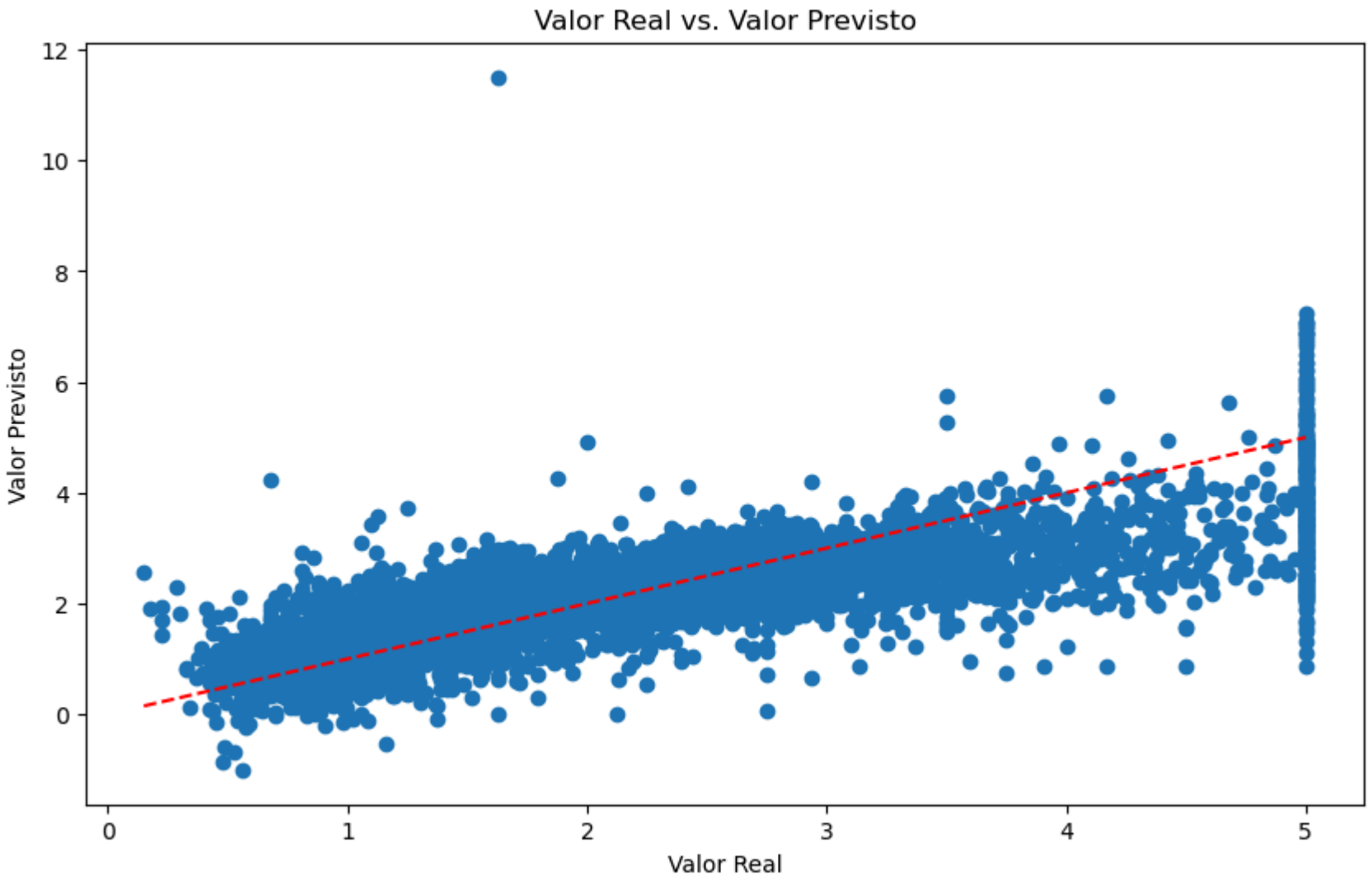
LinearRegression

LinearRegression()

In [59]:

```
y_pred = model.predict(X_test)
# Cálculo de métricas
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse:.2f}')
print(f'R²: {r2:.2f}')
# Visualização
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.xlabel('Valor Real')
plt.ylabel('Valor Previsto')
plt.title('Valor Real vs. Valor Previsto')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--') # linha de 45 graus
plt.show()
```

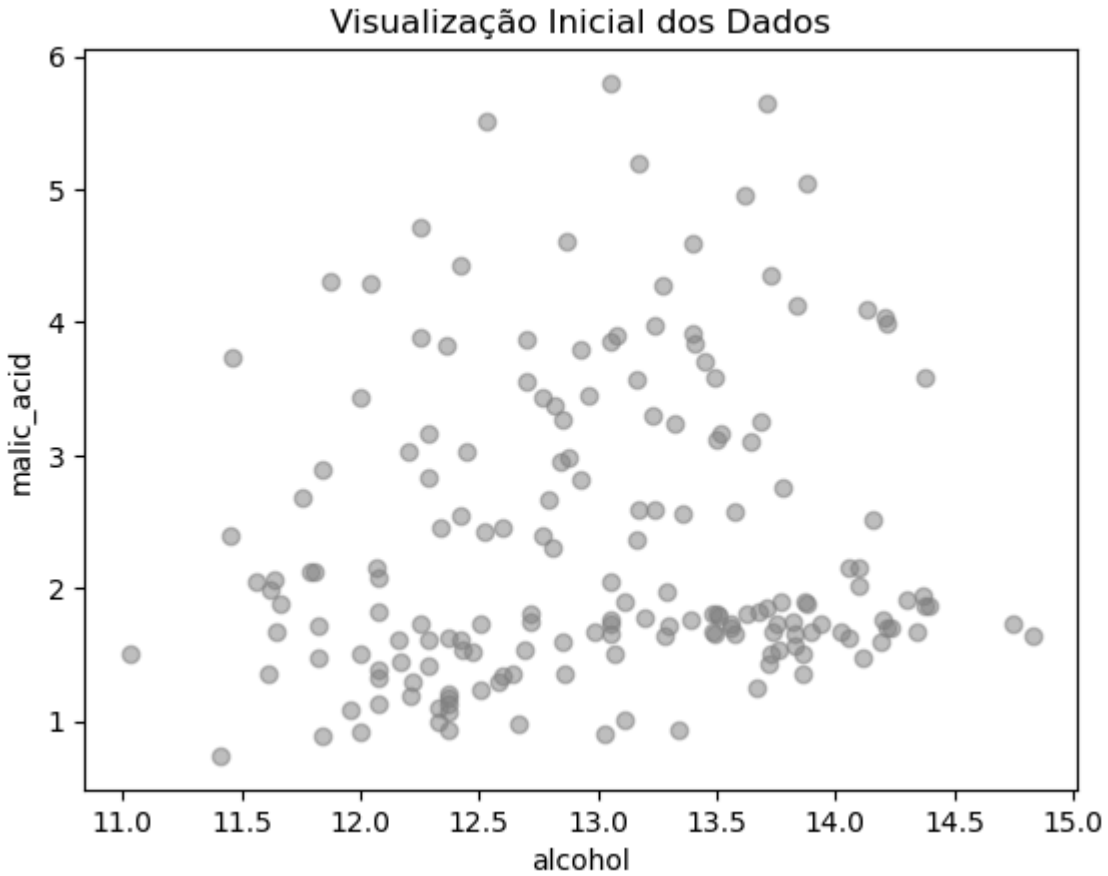
```
Mean Squared Error: 0.53
R²: 0.60
```



In [84]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

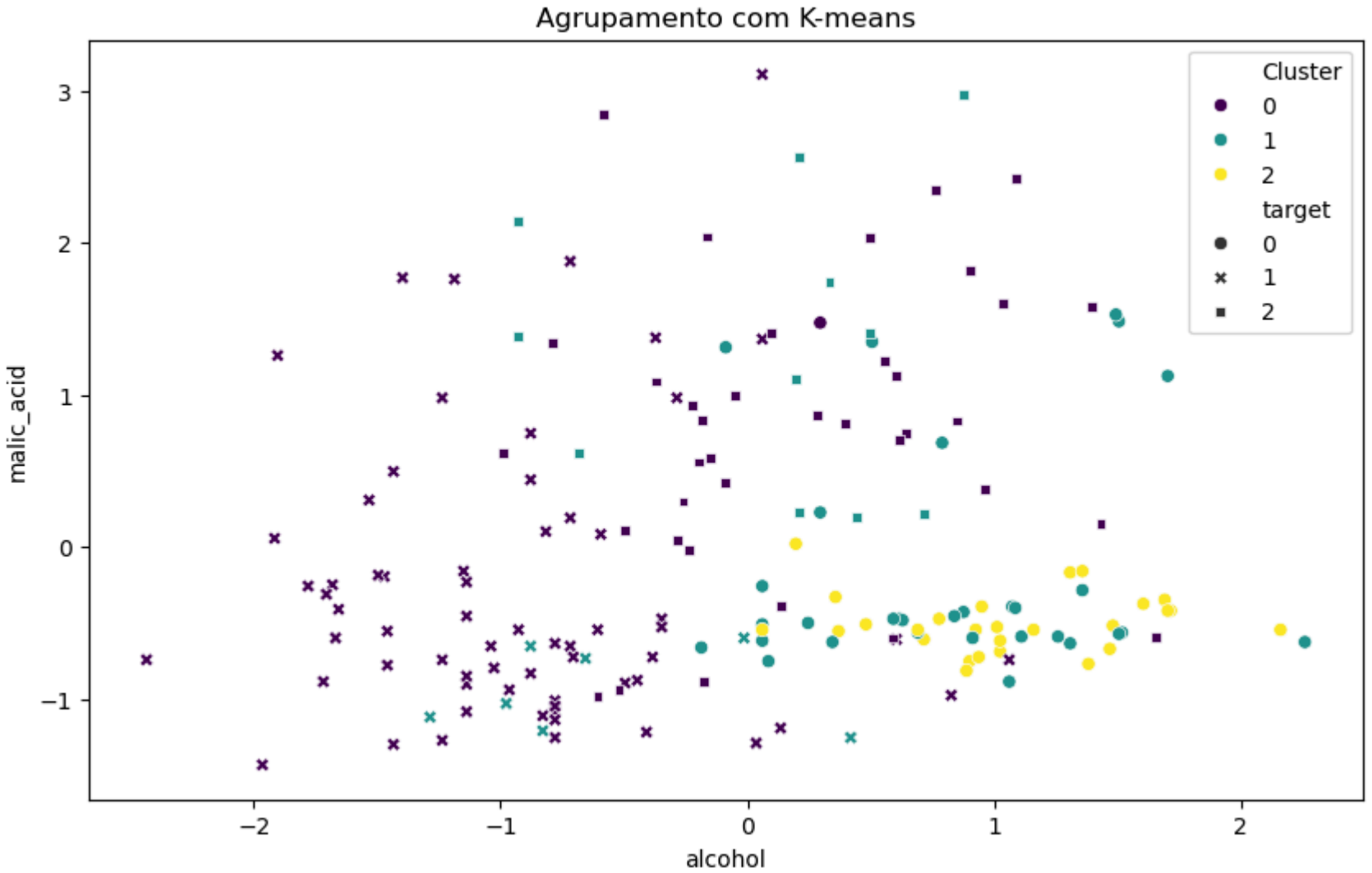
In [82]:
```
 data = load_wine()
X = data.data
# Visualizar os dados
plt.scatter(X[:, 0], X[:, 1], c='gray', alpha=0.5)
```

```
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
plt.title('Visualização Inicial dos Dados')
plt.show()
```



In [90]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df[data.feature_names[0]],
y=df[data.feature_names[1]],
hue='Cluster',
palette='viridis',
style='target',
data=df)
plt.title('Agrupamento com K-means')
```

```
plt.xlabel(data.feature_names[0])
10
plt.ylabel(data.feature_names[1])
plt.legend()
plt.show()
```



In [92]:
```
# Cálculo da silhueta
silhouette_avg = silhouette_score(X, clusters)
```

```
print(f'Silhouette Score: {silhouette_avg:.2f}')
```

```
Silhouette Score: 0.56
```

In [ ]:

In [ ]: