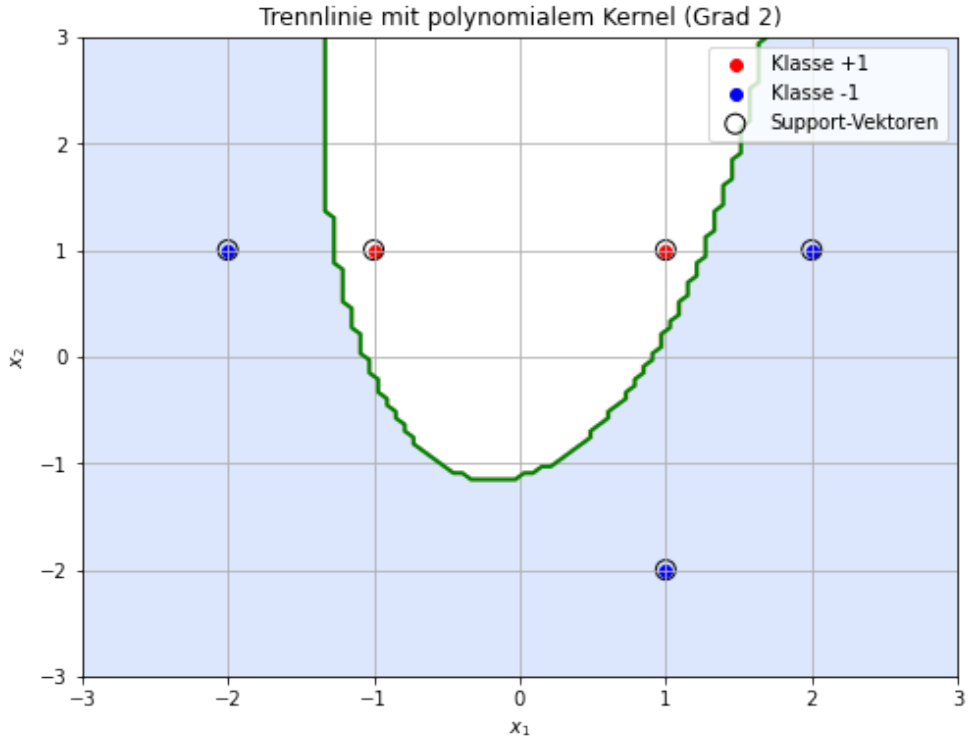
In [3]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
import time

# Definierte Datenpunkte und Labels
X = np.array([[1, 1], [-1, 1], [2, 1], [1, -2], [-2, 1]])  # A, B, C, D, E
y = np.array([1, 1, -1, -1, -1])  # Labels: +1 für A und B, -1 für C, D und E

# Erstelle das SVM-Modell mit einem polynomialen Kernel (Grad 2)
svm_model = SVC(kernel='poly', degree=2, coef0=1)  # coef0=1 entspricht dem k
onstanten Term in K
svm_model.fit(X, y)

# Bereich für x1 und x2 (damit die Trennlinie im Plot dargestellt wird)
x1_range = np.linspace(-3, 3, 100)
x2_range = np.linspace(-3, 3, 100)
xx1, xx2 = np.meshgrid(x1_range, x2_range)
grid = np.c_[xx1.ravel(), xx2.ravel()]

# Vorhersagen für das Gitter
Z = svm_model.predict(grid)
Z = Z.reshape(xx1.shape)

# Plot der Datenpunkte und der Entscheidungsfläche
plt.figure(figsize=(8, 6))

# Plot der Entscheidungsebene (Trennfläche)
plt.contourf(xx1, xx2, Z, alpha=0.3, cmap='coolwarm', levels=np.arange(-1.5,
1.5, 1))

# Grüne Trennlinie einzeichnen
plt.contour(xx1, xx2, Z, colors='green', levels=[0], linewidths=2)

# Plot der Datenpunkte
plt.scatter(X[y == 1, 0], X[y == 1, 1], color='red', label='Klasse +1')
plt.scatter(X[y == -1, 0], X[y == -1, 1], color='blue', label='Klasse -1')

# Plot der Support-Vektoren
plt.scatter(svm_model.support_vectors_[:, 0], svm_model.support_vectors_[:,
1],
            s=100, facecolors='none', edgecolors='k', label='Support-Vektore
n')

# Labels und Titel
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.title('Trennlinie mit polynomialem Kernel (Grad 2)')
plt.legend()
plt.grid(True)

# Zeige den Plot
plt.show()

# Berechne die Kernel-Matrix für die Datenpunkte
def polynomial_kernel(x, y, degree=2, coef0=1):
    return (np.dot(x, y) + coef0) ** degree

# Kernel-Matrix berechnen
kernel_matrix = np.zeros((X.shape[0], X.shape[0]))
for i in range(X.shape[0]):
    for j in range(X.shape[0]):
```

```
        kernel_matrix[i, j] = polynomial_kernel(X[i], X[j])

# Kernel-Matrix ausgeben
print("Kernel-Matrix:")
print(kernel_matrix)
```




```
Kernel-Matrix:
[[ 9.  1. 16.  0.  0.]
 [ 1.  9.  0.  4. 16.]
 [16.  0. 36.  1.  4.]
 [ 0.  4.  1. 36.  9.]
 [ 0. 16.  4.  9. 36.]]
```

In [4]:
```
# print current date and time

print("Date & Time:",time.strftime("%d.%m.%Y  %H:%M:%S"))
# end of import test
print ("*** End of SVM_PolyKernel(2)-5DP Jupyter Notebook ***")
```

```
Date & Time: 20.10.2024  20:42:08
*** End of SVM_PolyKernel(2)-5DP Jupyter Notebook ***
```