In [28]:
```
#importando librerias:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [29]:
```
# Cargando los datos:
train = pd.read_csv('./water_pollution.csv')
```

In [30]:
```
# Visualizando los datos:
train.head(3)
```

Out[30]:

| | City | Region | Country | AirQuality | WaterPollution |
|---|---|---|---|---|---|
| 0 | New York City | "New York" | "United States of America" | 46.816038 | 49.504950 |
| 1 | Washington, D.C. | "District of Columbia" | "United States of America" | 66.129032 | 49.107143 |
| 2 | San Francisco | "California" | "United States of America" | 60.514019 | 43.000000 |

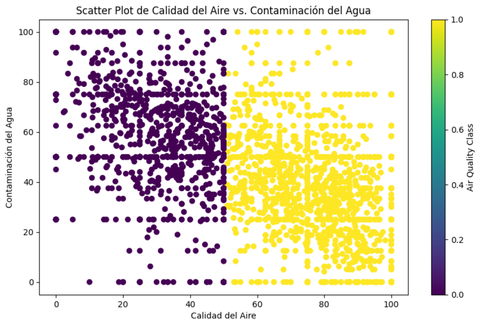
In [31]:
```
# Exploración de datos:
train[['AirQuality', 'WaterPollution']].head()
```

Out[31]:

| | AirQuality | WaterPollution |
|---|---|---|
| 0 | 46.816038 | 49.504950 |
| 1 | 66.129032 | 49.107143 |
| 2 | 60.514019 | 43.000000 |
| 3 | 62.364130 | 28.612717 |
| 4 | 36.621622 | 61.299435 |

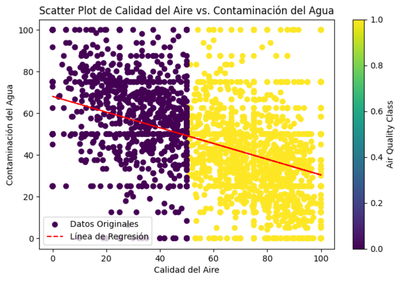
In [32]:
```
# Definir una variable binaria para clasificar la calidad del aire como buena (0) o mala (1)
train['AirQualityClass'] = (train['AirQuality'] > 50).astype(int)
```

In [33]:
```
# Graficar la variable dependiente e independiente
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.scatter(train['AirQuality'], train['WaterPollution'], c=train['AirQualityClass'], cmap='viridis')
plt.title('Scatter Plot de Calidad del Aire vs. Contaminación del Agua')
plt.xlabel('Calidad del Aire')
plt.ylabel('Contaminación del Agua')
plt.colorbar(label='Air Quality Class')
plt.show()
```




In [34]:
```
# Calcular el coeficiente de correlación de Pearson
correlation = train['AirQuality'].corr(train['WaterPollution'])

# Imprimir el coeficiente de correlación
print(f'Coeficiente de correlación de Pearson: {correlation}')

# Graficar la relación entre la calidad del aire y la contaminación del agua
plt.figure(figsize=(8, 5))
plt.scatter(train['AirQuality'], train['WaterPollution'], c=train['AirQualityClass'], cmap='viridis', label='Datos Originales')
plt.title('Scatter Plot de Calidad del Aire vs. Contaminación del Agua')
plt.xlabel('Calidad del Aire')
plt.ylabel('Contaminación del Agua')
plt.colorbar(label='Air Quality Class')
plt.legend()

# Agregar una línea de regresión para visualizar la tendencia
z = np.polyfit(train['AirQuality'], train['WaterPollution'], 1)
p = np.poly1d(z)
plt.plot(train['AirQuality'], p(train['AirQuality']), 'r--', label='Línea de Regresión')

plt.legend()
plt.show()
```

```
Coeficiente de correlación de Pearson: -0.45417262259393115
```




In [35]:
```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train = train[['AirQuality', 'WaterPollution']]
y_train = train['AirQualityClass']
```

In [36]:
```
# Inicializar el modelo de regresión logistica
model = LogisticRegression()
```

In [37]:
```
# Entrenar el modelo
model.fit(X_train, y_train)
```

Out[37]:
```
LogisticRegression
LogisticRegression()
```

In [38]:
```
# Realizar predicciones en el conjunto de prueba
y_pred = model.predict(X_train)
```

In [39]:
```
# Evaluar el rendimiento del modelo en el conjunto de entrenamiento
accuracy = accuracy_score(y_train, y_pred)
conf_matrix = confusion_matrix(y_train, y_pred)
class_report = classification_report(y_train, y_pred)

print(f'Accuracy en el conjunto de entrenamiento: {accuracy}')
print(f'Confusion Matrix en el conjunto de entrenamiento:\n{conf_matrix}')
print(f'Classification Report en el conjunto de entrenamiento:\n{class_report}')
```

```
Accuracy en el conjunto de entrenamiento: 0.9992429977289932
Confusion Matrix en el conjunto de entrenamiento:
[[1449    0]
 [   3 2511]]
Classification Report en el conjunto de entrenamiento:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1449
           1       1.00      1.00      1.00      2514

    accuracy                           1.00      3963
   macro avg       1.00      1.00      1.00      3963
weighted avg       1.00      1.00      1.00      3963
```

In [40]:
```
# Supongamos que estos son los valores de tu nuevo dato
new_data = {
    'City': ["New City"],
    'Region': ["New Region"],
    'Country': ["New Country"],
    'AirQuality': [65],
    'WaterPollution': [30]
}

# Crea un DataFrame con el nuevo dato
new_df = pd.DataFrame(new_data)

# Aplica el mismo preprocesamiento que hiciste con el conjunto de entrenamiento
new_df['AirQualityClass'] = (new_df['AirQuality'] > 50).astype(int)

# Selecciona las características para la predicción
X_new = new_df[['AirQuality', 'WaterPollution']]

# Realiza la predicción con el modelo entrenado
prediction = model.predict(X_new)

print(f'Predicted Air Quality Class for the new data: {prediction}')
```

```
Predicted Air Quality Class for the new data: [1]
```

In [41]:
```
# Graficar la variable dependiente e independiente
plt.figure(figsize=(10, 6))

# Scatter plot de los datos originales
plt.scatter(train['AirQuality'], train['WaterPollution'], c=train['AirQualityClass'], cmap='viridis', label='Datos Originales')

# Scatter plot de los nuevos datos para predicción
plt.scatter(new_df['AirQuality'], new_df['WaterPollution'], c='red', marker='x', label='Nuevo Dato para Predicción')

plt.title('Scatter Plot de Calidad del Aire vs. Contaminación del Agua')
plt.xlabel('Calidad del Aire')
plt.ylabel('Contaminación del Agua')
plt.colorbar(label='Air Quality Class')

# Agregar una línea de regresión para visualizar la tendencia
z = np.polyfit(train['AirQuality'], train['WaterPollution'], 1)
p = np.poly1d(z)
plt.plot(train['AirQuality'], p(train['AirQuality']), 'r--', label='Línea de Regresión')

plt.legend()
plt.show()
```