In [55]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

In [57]:
```
df = pd.read_csv("Nat_Gas.csv")
```

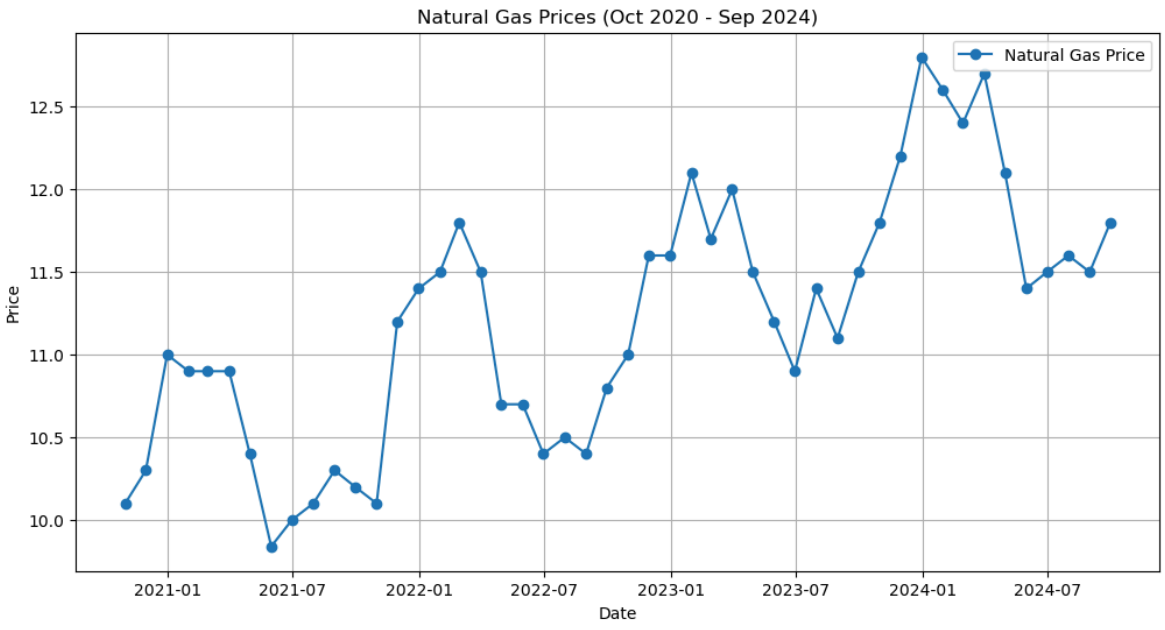
In [59]:
```
df.head()
```

Out[59]:

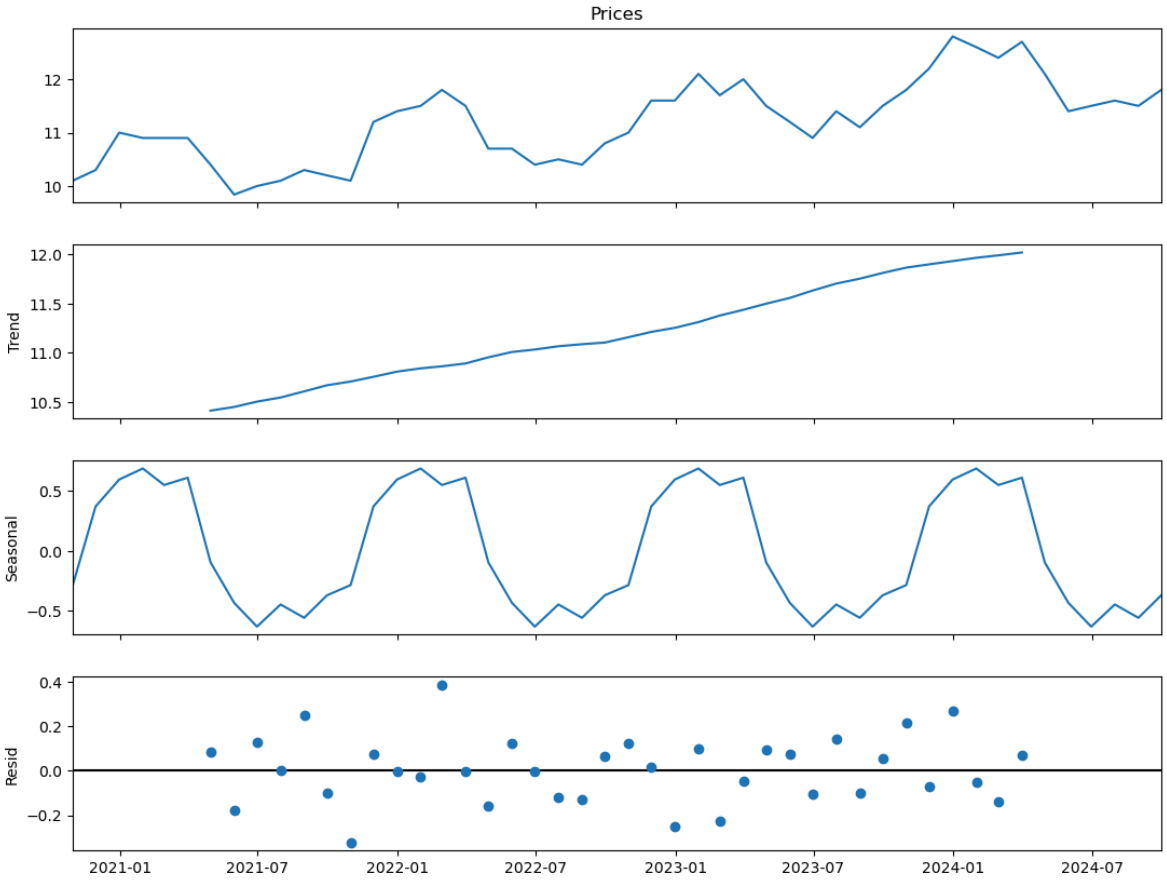
| | Dates | Prices |
|---|---|---|
| **0** | 10/31/20 | 10.1 |
| **1** | 11/30/20 | 10.3 |
| **2** | 12/31/20 | 11.0 |
| **3** | 1/31/21 | 10.9 |
| **4** | 2/28/21 | 10.9 |

In [61]:
```
# Ensure Dates column exists and is datetime
df.reset_index(inplace=True)   # in case Dates was already index
df['Dates'] = pd.to_datetime(df['Dates'], format="%m/%d/%y")
df.set_index('Dates', inplace=True)   # now index is proper datetime
```

In [63]:
```
# --- Visualization ---
plt.figure(figsize=(12,6))
plt.plot(df.index, df['Prices'], marker='o', linestyle='-', label="Natural Gas P
plt.title("Natural Gas Prices (Oct 2020 - Sep 2024)")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.grid(True)
plt.show()
```



In [65]:
```
# --- Seasonal Decomposition ---
decomposition = seasonal_decompose(df['Prices'], model='additive', period=12)
fig = decomposition.plot()
fig.set_size_inches(12, 9)
plt.show()
```

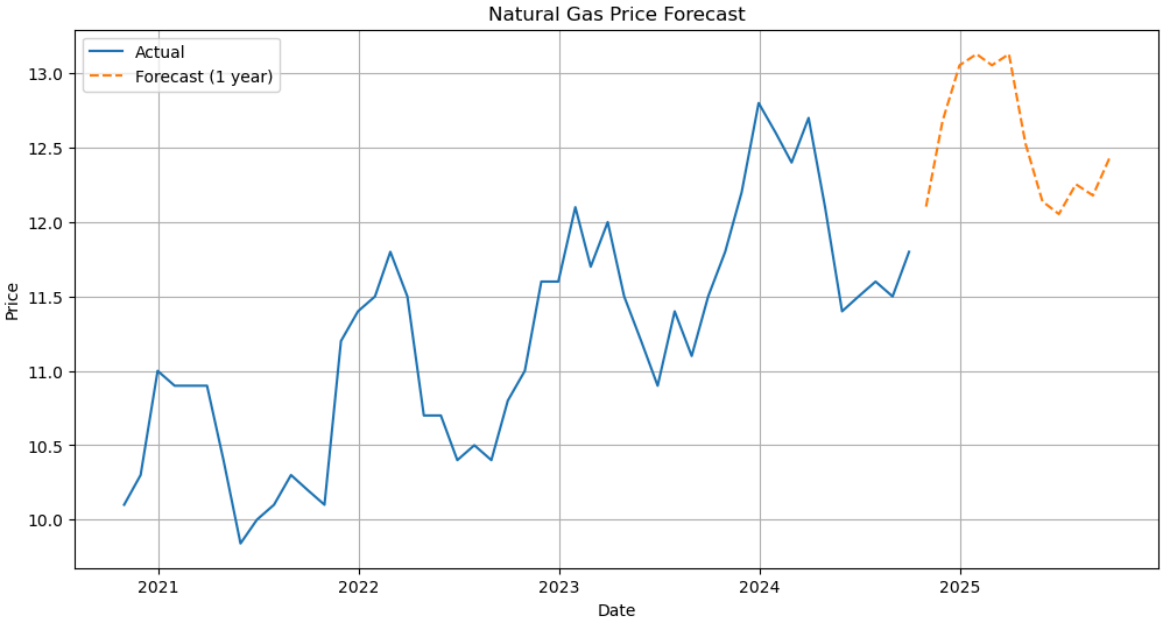



In [67]:
```
# --- Forecasting Model ---
model = ExponentialSmoothing(df['Prices'], trend="add", seasonal="add", seasonal
fit = model.fit()
```

```
C:\Users\user\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency M will
be used.
  self._init_dates(dates, freq)
```

In [69]:
```
# Forecast 12 months ahead
forecast = fit.forecast(12)
```

In [71]:
```
# Plot actual vs forecast
plt.figure(figsize=(12,6))
plt.plot(df.index, df['Prices'], label="Actual")
plt.plot(forecast.index, forecast, label="Forecast (1 year)", linestyle="--")
plt.title("Natural Gas Price Forecast")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.grid(True)
plt.show()
```




In [87]:
```
def estimate_price(input_date):
    date = pd.to_datetime(input_date)

    if date in df.index:
        return float(df.loc[date, "Prices"])

    elif date > df.index[-1]:
        steps_ahead = (date.to_period("M") - df.index[-1].to_period("M")).n
        # Forecast as many months ahead as needed
        return float(fit.forecast(steps_ahead).iloc[-1])

    else:
        return float(np.interp(
            date.toordinal(),
            df.index.to_series().map(pd.Timestamp.toordinal),
            df['Prices']
        ))
```

In [89]:
```
print("Price on 2020-10-31:",estimate_price("2020-10-31"))
print("Price on 2026-10-31:", estimate_price("2026-10-31"))
```

```
Price on 2020-10-31: 10.1
Price on 2026-10-31: 13.186818307897802
```

In [ ]: