In [288]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [289]:
```
data = pd.read_csv('infolimpioavanzadoTarget.csv',usecols=['open','high','low',
```

In [290]:
```
data.head(5)
```

Out[290]:

	open	high	low	close	adjclose	volume
0	17.799999	18.219000	17.500000	17.760000	17.760000	106600
1	17.700001	18.309999	17.620001	17.660000	17.660000	128700
2	17.580000	17.799999	16.910000	16.950001	16.950001	103100
3	16.650000	16.879999	16.139999	16.170000	16.170000	173600
4	16.219999	16.290001	15.630000	15.710000	15.710000	137800

In [291]:
```
data.isnull().sum()
```

Out[291]:
```
open        0
high        0
low         0
close       0
adjclose    0
volume      0
dtype: int64
```

In [292]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7781 entries, 0 to 7780
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   open      7781 non-null   float64
 1   high      7781 non-null   float64
 2   low       7781 non-null   float64
 3   close     7781 non-null   float64
 4   adjclose  7781 non-null   float64
 5   volume    7781 non-null   int64  
dtypes: float64(5), int64(1)
memory usage: 364.9 KB
```

In [293]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7781 entries, 0 to 7780
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   open      7781 non-null   float64
 1   high      7781 non-null   float64
 2   low       7781 non-null   float64
 3   close     7781 non-null   float64
 4   adjclose  7781 non-null   float64
 5   volume    7781 non-null   int64  
dtypes: float64(5), int64(1)
memory usage: 364.9 KB
```

In [294]:
```
predict_days =60
```

In [295]:
```
data['predicted'] = data['adjclose'].shift(-predict_days)
```

In [296]:
```
x = np.array(data.drop(['predicted'],axis=1))
x = x[:-predict_days]
```

In [303]:
```
y = np.array(data['predicted'])
y = y[:-predict_days]
```

In [304]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [305]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(5404, 6)
(2317, 6)
(5404,)
(2317,)
```

In [306]:
```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
```

In [307]:
```
ridge_model = Ridge()
ridge_model.fit(x_train, y_train)

ridge_model_score = ridge_model.score(x_test, y_test)
print('Ridge Model score:', ridge_model_score)
```

```
Ridge Model score: 0.5385512514935036
```

In [308]:
```
x_predict = np.array(data.drop(['predicted'], 1))[-predict_days:]
ridge_model_predict_prediction = ridge_model.predict(x_predict)
ridge_model_real_prediction = ridge_model.predict(np.array(data.drop(['predicte
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_10164\955829814.py:1: FutureWarnin
g: In a future version of pandas all arguments of DataFrame.drop except for th
e argument 'labels' will be keyword-only.
  x_predict = np.array(data.drop(['predicted'], 1))[-predict_days:]
C:\Users\DELL\AppData\Local\Temp\ipykernel_10164\955829814.py:3: FutureWarnin
g: In a future version of pandas all arguments of DataFrame.drop except for th
e argument 'labels' will be keyword-only.
  ridge_model_real_prediction = ridge_model.predict(np.array(data.drop(['predi
cted'], 1)))
```

In [326]:
```
# Assuming 'date' is  index column, and it needs to be converted to datetime
data = pd.read_csv('infolimpioavanzadoTarget.csv',usecols=['date','open','high'
data['date'] = pd.to_datetime(data['date'])
data.set_index('date', inplace=True)

# Defining some Parameters
from datetime import timedelta

predicted_dates = []
recent_date = data.index.max()
display_at = 1000
alpha = 0.5
predict_days = 10

for i in range(predict_days):
    recent_date = recent_date + timedelta(days=1)
    predicted_dates.append(recent_date)

# Displaying the first few predicted dates
print(predicted_dates[:display_at])
```

```
[Timestamp('2022-12-31 00:00:00'), Timestamp('2023-01-01 00:00:00'), Timestamp
('2023-01-02 00:00:00'), Timestamp('2023-01-03 00:00:00'), Timestamp('2023-01-
04 00:00:00'), Timestamp('2023-01-05 00:00:00'), Timestamp('2023-01-06 00:00:0
0'), Timestamp('2023-01-07 00:00:00'), Timestamp('2023-01-08 00:00:00'), Times
tamp('2023-01-09 00:00:00')]
```

In [327]:
```
lasso_model = Lasso()
lasso_model.fit(x_train, y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_de
scent.py:631: ConvergenceWarning: Objective did not converge. You might want t
o increase the number of iterations, check the scale of the features or consid
er increasing regularisation. Duality gap: 7.949e+06, tolerance: 5.210e+03
  model = cd_fast.enet_coordinate_descent(
```

Out[327]:

▾ Lasso

Lasso()

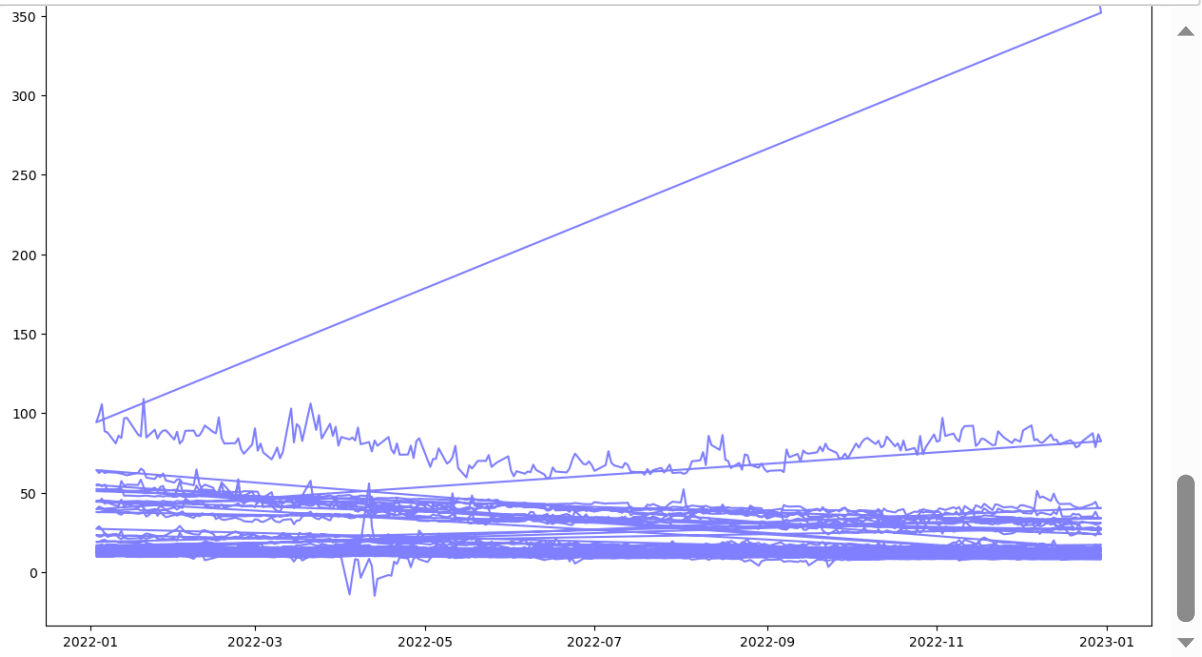
In [328]:
```
# Score of the Lasso Regression Model (Using the Test Data)

lasso_model_score = lasso_model.score(x_test, y_test)
print('Lasso Model score:', lasso_model_score)
```

```
Lasso Model score: 0.5329250618941705
```

In [334]:
```
data['predicted'] = data['adjclose'].shift(-predict_days)
ridge_model_predict_prediction = ridge_model.predict(x_predict)
ridge_model_real_prediction = ridge_model.predict(np.array(data.drop(['predicte
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_10164\1654179197.py:3: FutureWarnin
g: In a future version of pandas all arguments of DataFrame.drop except for th
e argument 'labels' will be keyword-only.
  ridge_model_real_prediction = ridge_model.predict(np.array(data.drop(['predi
cted'], 1)))
```

In [335]:
```
plt.figure(figsize=(15, 9))
plt.plot(data.index[display_at:], ridge_model_real_prediction[display_at:], lab
plt.plot(predicted_dates, ridge_model_predict_prediction, label='Forecast', col
plt.plot(data.index[display_at:], data['Close'][display_at:], label='Actual', c
plt.legend()
```



In [ ]: