In [1]:
```python
# Python
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from prophet import Prophet
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
import matplotlib.pyplot as plt
from prophet.plot import plot_cross_validation_metric
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
import funciones
import xgboost as xgb
```

In [2]:
```python
df_main = pd.read_excel("https://raw.githubusercontent.com/carrenogf/MCD-Series-
df_main = df_main.sort_values("FECHA",ascending=True)
df_main.set_index("FECHA", inplace=True)
df_copa = df_main["CHU_COPA_AJUST"].dropna()
df_recprop = df_main["CHU_REC_PROPIOS_AJUST"].dropna()
df_regal = df_main["CHU_REGALIAS_AJUST"].dropna()
dataframes = [df_copa, df_recprop, df_regal]
for i in range(len(dataframes)):
  dataframes[i] = dataframes[i].reindex(pd.date_range(start=dataframes[i].index.
  dataframes[i] = dataframes[i].fillna(0)

titulos = ["CHU_COPA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

In [3]:
```python
def extract_time_features(index):
    return pd.Series({
        'dayofweek': index.dayofweek,
        'quarter': index.quarter,
        'month': index.month,
        'year': index.year,
        'dayofyear': index.dayofyear,
        'dayofmonth': index.day,
        'weekofyear': index.isocalendar().week
    })
def add_lags(df, titulo):
    target_map = df[titulo].to_dict()
    df['lag1'] = (df.index - pd.Timedelta('364 days')).map(target_map) # df_1['P
    df['lag2'] = (df.index - pd.Timedelta('728 days')).map(target_map) # df_1['P
    df['lag3'] = (df.index - pd.Timedelta('1092 days')).map(target_map) # df_1['
    return df

for i in range(len(dataframes)):
    time_features = dataframes[i].index.to_series().apply(extract_time_features)
    dataframes[i] = pd.concat([dataframes[i], time_features], axis=1)
    dataframes[i] = add_lags(dataframes[i], titulos[i])
```

In [4]:
```python
# TRAIN TEST
n_test = 30
train_copa = dataframes[0].iloc[:-n_test]
test_copa = dataframes[0].iloc[-n_test:]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")
```

```
train_recursos = dataframes[1].iloc[:-n_test]
test_recursos = dataframes[1].iloc[-n_test:]
print(f"Recursos: train({train_recursos.shape}), test({test_recursos.shape})")

train_regalias = dataframes[2].iloc[:-n_test]
test_regalias = dataframes[2].iloc[-n_test:]
print(f"Regalias: train({train_regalias.shape}), test({test_regalias.shape})")

dataframes_train = [ train_copa, train_recursos, train_regalias ]
dataframes_test = [ test_copa, test_recursos, test_regalias ]
```

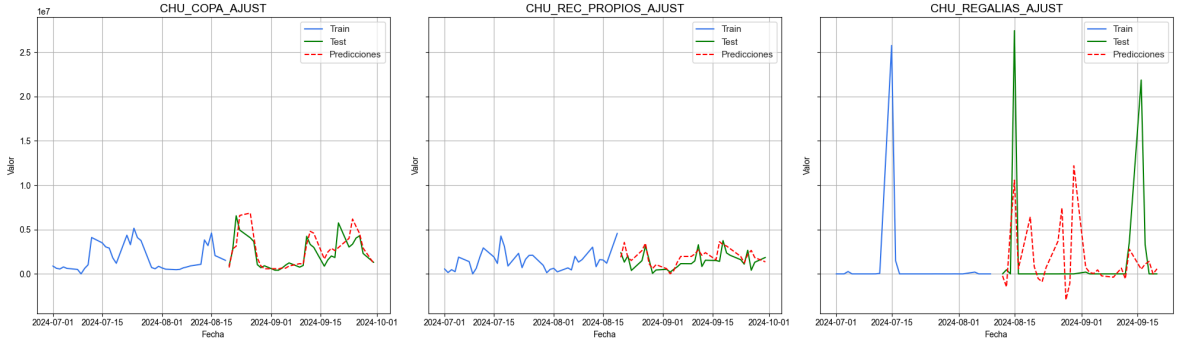
```
Coparticipacion: train((1730, 11)), test((30, 11))
Recursos: train((2187, 11)), test((30, 11))
Regalias: train((2176, 11)), test((30, 11))
```

In [ ]:

```
results_train_test = []
predictions_test = []
best_params = pd.read_csv("xgb_best_params.csv",sep=";")

for i, df_train in enumerate(dataframes_train):

    params = eval(best_params.iloc[i]["best_params"])

    FEATURES = ['dayofweek', 'quarter', 'month', 'year', 'dayofyear' , 'dayofmon
    TARGET = titulos[i]

    df_test = dataframes_test[i][TARGET]
    model = xgb.XGBRegressor(**params )

    X_train = dataframes_train[i][FEATURES]
    y_train = dataframes_train[i][TARGET]
    X_test= dataframes_test[i][FEATURES]
    y_test= dataframes_test[i][TARGET]

    model.fit(
        X_train, y_train,
        eval_set=[(X_test, y_test)]
    )

    pred_test = model.predict(X_test)
    pred_test = pd.Series(pred_test, index=dataframes_test[i].index)
    predictions_test.append(pred_test)

    # Cálculo del MSE en el conjunto de prueba
    mape_test = mean_absolute_percentage_error(df_test, pred_test)
    mape_mean = mean_absolute_percentage_error(df_test, [df_test.mean()] * len(d
    mse_test = mean_squared_error(df_test, pred_test)
    mae_test = mean_absolute_error(df_test, pred_test)
    rmse = np.sqrt(mean_squared_error(df_test, pred_test))
    results_train_test.append({
        "model": model,
        "name": titulos[i],
        "len_train": len(df_train),
        "len_test": len(df_test),
        "mape_test": mape_test,
        "mse_test":mse_test,
        "mape_mean": mape_mean,
        "mae_test": mae_test,
        "rmse": rmse
    })
```

In [6]:
```
pd.options.display.float_format = '{:,.2f}'.format
display(pd.DataFrame(results_train_test))

display(funciones.plot_train_test_predictions(
    dataframes_train=[pd.Series(dataframes_train[i][titulos[i]], index=dataframe
    dataframes_test=[pd.Series(dataframes_test[i][titulos[i]], index=dataframes_
    predictions_test=predictions_test,
    series_names=titulos,
    start_date='2024-07-01'
))
```

	model	name	len_train	len_test	
0	XGBRegressor(base_score=None, booster=None, ca...	CHU_COPA_AJUST	1730	30	
1	XGBRegressor(base_score=None, booster=None, ca...	CHU_REC_PROPIOS_AJUST	2187	30	
2	XGBRegressor(base_score=None, booster=None, ca...	CHU_REGALIAS_AJUST	2176	30	5,880,061,7



None

In [ ]: