In [1]:

```python
# Python
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from autots import AutoTS
import matplotlib.pyplot as plt
import funciones
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
```

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
# 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,)), test((30,))
Recursos: train((2187,)), test((30,))
Regalias: train((2176,)), test((30,))
```

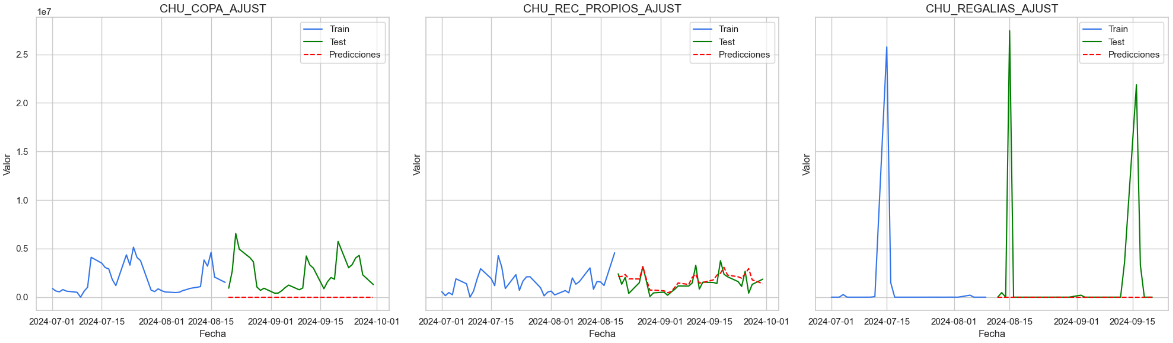
In [ ]:

```python
results_train_test = []
predictions_test = []
for i, df in enumerate(dataframes_train):
    df_train = df
    df_test = dataframes_test[i]

    model = AutoTS(
        forecast_length=len(dataframes_test[i]),
        frequency="B",
        prediction_interval=0.95,
        ensemble=None,
        models_mode='deep',
        model_list = 'superfast',
        max_generations=10,   # intenta optimizar el modelo a traves de 10 itera
        num_validations=3,
        no_negatives=True,
        n_jobs='auto')
    modelAutoTS = model.fit(df_train)
    # Find the best parameters

    fechas = pd.date_range(start=df_test.index.min(), end=df_test.index.max(), f
    pred_test = model.predict(forecast_length=len(fechas)).forecast
    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": modelAutoTS,
        "name": df_train.name,
        "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 [7]:

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

display(funciones.plot_train_test_predictions(
    dataframes_train=dataframes_train,
    dataframes_test=dataframes_test,
    predictions_test=predictions_test,
    series_names=titulos,
    start_date='2024-07-01'
))
```

	model	name	len_train	len_test	mape_test	mse_test
0	Initiated AutoTS object with best model: \nLas...	CHU_COPA_AJUST	1730	30	1.00	8,545,162,889,493.53
1	Initiated AutoTS object with best model: \nSea...	CHU_REC_PROPIOS_AJUST	2187	30	0.99	547,894,342,942.74
2	Initiated AutoTS object with best model: \nSea...	CHU_REGALIAS_AJUST	2176	30	0.27	41,804,929,164,773.93



```
None
```

In [6]:

```python
results = pd.DataFrame(results_train_test)

for i, row in results.iterrows():
    display(row["name"])
    print(row.model)
```

```
'CHU_COPA_AJUST'
```

```
Initiated AutoTS object with best model:
LastValueNaive
{'fillna': 'quadratic', 'transformations': {'0': 'RollingMeanTransformer', '1':
'AnomalyRemoval', '2': 'bkfilter', '3': 'ScipyFilter'}, 'transformation_params':
{'0': {'fixed': True, 'window': 360, 'macro_micro': True, 'center': True}, '1':
{'method': 'zscore', 'method_params': {'distribution': 'uniform', 'alpha': 0.05},
'fillna': 'ffill', 'transform_dict': {'fillna': None, 'transformations': {'0': 'E
WMAFilter'}, 'transformation_params': {'0': {'span': 7}}}, 'isolated_only': Fals
e}, '2': {}, '3': {'method': 'butter', 'method_args': {'N': 2, 'btype': 'highpas
s', 'analog': False, 'output': 'sos', 'Wn': 0.015384615384615385}}}}
{}
Validation: 0, 1, 2, 3
SMAPE: 80.45022147190438, nan, nan, nan
MAE: 1457881.601770488, 1754848.8054795982, 3351528.787366555, 1944707.5176069036
SPL: 0.18310796907270413, 0.3516077000937226, 1.516704946141438, 0.23639689726808
616
'CHU_REC_PROPIOS_AJUST'
Initiated AutoTS object with best model:
SeasonalityMotif
{'fillna': 'akima', 'transformations': {'0': 'PositiveShift', '1': 'Detrend'}, 't
ransformation_params': {'0': {}, '1': {'model': 'GLS', 'phi': 1, 'window': 365,
'transform_dict': None}}}
{'window': 15, 'point_method': 'midhinge', 'distance_metric': 'mae', 'k': 20, 'da
tepart_method': 'simple', 'independent': False}
Validation: 0, 1, 2, 3
SMAPE: 46.63511981455654, 81.92051984679205, 67.9873722891953, 59.817692083104504
MAE: 672472.5785951081, 1013335.51811632, 1024976.4424628529, 665533.8818280211
SPL: 0.14172371881288814, 0.22842532323077214, 0.16493539716519867, 0.16548226664
122395
'CHU_REGALIAS_AJUST'
Initiated AutoTS object with best model:
SeasonalityMotif
{'fillna': 'ffill', 'transformations': {'0': 'CenterLastValue', '1': 'QuantileTra
nsformer'}, 'transformation_params': {'0': {'rows': 6}, '1': {'output_distributio
n': 'uniform', 'n_quantiles': 1000}}}
{'window': 15, 'point_method': 'midhinge', 'distance_metric': 'mse', 'k': 20, 'da
tepart_method': 'simple', 'independent': True}
Validation: 0, 1, 2, 3
SMAPE: nan, nan, nan, nan
MAE: 926159.4666666667, 1861666.2666666666, 1564803.1333333333, 2344361.266666666
6
SPL: 0.19809204672333816, 0.5484111223643126, 0.21565462335433694, 0.159545677128
9861
```

```
In [ ]:
```