

**TRABAJO FIN DE MÁSTER: INTELIGENCIA ARTIFICIAL APLICADA A LA GESTIÓN DE RECURSOS DE SISTEMAS TRANSACCIONALES.**

- Universidad Internacional de La Rioja (UNIR) | Escuela Superior de Ingeniería y Tecnología.
- Máster Universitario en Análisis y Visualización de Datos Masivos.

Presentado por: **Diego Fernando Gamboa Gómez**

Director/a: **María de la Luz Morales Jiménez**

Para la realización de este trabajo se ha utilizado el lenguaje de programación Python.

A lo largo del documento hay referencias a guías, repositorios y artículos con información y conceptos claves que sirvieron de base para desarrollo de funciones y modelos, al igual que para el análisis e interpretación de los resultados.

## **1 Configuración de ambiente de trabajo**

### **1.1 Librerias Python**

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sys
import os

plt.rcParams['figure.figsize'] = (15, 5)
sns.set_style('whitegrid')
```

## 1.2 Opciones de pandas

In [2]:

```
pd.set_option('display.max_columns', 30)
pd.set_option('display.max_colwidth', 150)
pd.set_option("display.precision", 3)
pd.set_option('display.max_rows', 100)
```

## 2 Funciones

In [3]:

```
# -----
# Función para eliminar columnas del dataframe si existen.
# -----
def delete_columns(_df, del_cols=[]):
    del_cols = [col for col in no_utiles if col in _df.columns]
    if set(del_cols).issubset(_df.columns):
        _df.drop(del_cols, axis=1, inplace=True )
        print("Se eliminaron satisfactoriamente las columnas del dataset.")
        print("", "")
        print("Las siguientes campos no existen como columnas en el dataset:\n", "-"*30, "\n"
              + str([col for col in no_utiles if col not in _df.columns]))
    )
    return _df

# -----
# Función para remover posibles outliers
# -----
from scipy.stats import zscore
def clean_outliers_zscore(df, columns=None, n=3, show_outliers=False):
    '''As seen at:
       <https://stackoverflow.com/questions/62366263/detect-the-outliers>
       <https://www.kite.com/python/answers/how-to-remove-outliers-from-a-pandas-dataframe-in-python>
    '''

    n_rows, _ = df.shape
    if not columns:
        columns = list(df.columns)

    dtos_nums = list(df[columns].select_dtypes(exclude = ['datetime', 'object']).columns)
    z_scores = zscore(df[dtos_nums])
    abs_z_scores = np.abs(z_scores)
    filtered_entries = (abs_z_scores < n).all(axis=1)
    outliers_df = df[~filtered_entries]
    new_df = df[filtered_entries]
    n_rows2, _ = new_df.shape

    if show_outliers:
        print("\nOutliers describe")
        display(outliers_df.describe())

    print("Registros Eliminados", n_rows - n_rows2)
    return new_df
```

```
# -----  
# Check unique - Determina si el valor de la columna es el mismo en todos los registros (filas)  
# -----  
def check_unique(u_df):  
    """  
        Función para determinar si alguna de las columnas que componen la fuente  
        de datos tiene el mismo valor en todos los registros.  
    """  
  
    unique_cols = []  
    elms = 0  
    for col in list(u_df.columns):  
        elms = len(pd.unique(u_df[col]))  
        if elms <= 1:  
            unique_cols.append(col)  
        elms = 0  
    if len(unique_cols) < 1 :  
        unique_cols.append("DataFrame no tiene columnas con valores únicos")  
  
    return unique_cols
```

In [4]:

```
# -----  
# Inspection dataframe - Inspección del dataframe  
# -----  
import dataframe_image as dfi  
from datetime import date  
  
def inspection_df(df):  
  
    # Número de filas y columnas en el dataframe  
    display(df.shape)  
    print("\n")  
  
    # Ver detalles estadísticos básicos  
    if len(df.select_dtypes(include=['object']).columns) > 0:  
        print(df.select_dtypes(include=['object']))  
        display(df.describe(include=np.object))  
    else:  
        display(df.describe())  
    print("\n")  
  
    # Nombre de las columnas en el dataframe  
    display(list(df.columns))  
    print("\n")  
  
    # Tipo de datos en cada columna del dataframe  
    display(df.dtypes)  
    print("\n")  
  
    # Imprimir un resumen del DataFrame  
    # índice y los tipos de columna, los valores no nulos y el uso de memoria.  
    display(df.info())  
    print("\n")  
  
# -----  
# Pairplot  
# -----  
def min_pairplot(df):  
    cls_df = list(df.select_dtypes(exclude=['datetime', 'object']).columns)  
    n_ctrl1 = int(len(cls_df)/2)  
  
    cls_df = [cls_df[:n_ctrl1], cls_df[n_ctrl1:]]  
  
    for cls in cls_df:
```

```

print(cls)
sns.pairplot(df[cls].select_dtypes(
    exclude=['datetime', 'object']), height=2.5)
plt.show()

# -----
# Confusion matrix Metrics
# https://stackoverflow.com/questions/31324218/scikit-learn-how-to-obtain-true-positive-true-negative-false-positive-and-fal
# -----


def show_mc_mtr(y_validation, predictions, dsp=None):
    """
    Calcula las métricas de una matriz de confusión.
    As seen at: https://bit.ly/3C86un2
    """

    from mlxtend.plotting import plot_confusion_matrix
    c_mt = confusion_matrix(y_validation, predictions)

    TP = np.diag(c_mt) # Diagonal
    rt = c_mt.sum(axis=1) # Suma Filas (row) [ Positivos: (TP + Fp) ]
    ct = c_mt.sum(axis=0) # Suma columnas (column)
    tm = np.sum(c_mt) # Suma toda La matriz
    tm = c_mt.sum() # Suma Matriz

    FP = [ct[i] - TP[i] for i in range(len(TP))]
    FN = [rt[i] - TP[i] for i in range(len(TP))]
    TN = [tm - (TP[i]+FP[i]+FN[i]) for i in range(len(TP))]

    resul = {}
    for i in range(len(TP)):
        resul[f'Clase_{i}'] = {
            'TP': TP[i],
            'FP': FP[i],
            'FN': FN[i],
            'TN': TN[i],
            # Sensibilidad, tasa de aciertos, recuperación o tasa de verdaderos positivos
            'Recall': round(TP[i]/(TP[i]+FN[i]), 3),
            # Especificidad o tasa negativa verdadera
            'Specificity': round(TN[i]/(TN[i]+FP[i]), 3),
            # Precisión o valor predictivo positivo
            'PositivePredictiveValue': round(TP[i]/(TP[i]+FP[i]), 3),
            # Valor predictivo negativo
            'NegativePredictiveValue': round(TN[i]/(TN[i]+FN[i]), 3),
        }

```

```

# Tasa de falsos positivos
'FalsePositiveRate': round(FP[i]/(FP[i]+TN[i]), 3),
# Tasa de falsos negativos
'FalseNegativeRate': round(FN[i]/(TP[i]+FN[i]), 3),
# Accuracy
'Accuracy': round((TP[i]+TN[i])/tm, 3),
# Error Rate
'ErrorRate': round((1-(TP[i]+TN[i])/tm), 3)
}

if dsp == 'dict':
    # Muestra Resultado (Dict)
    display(resul)
elif dsp == 'df':
    # Muestra Resultado (dataframe)
    mfg = pd.DataFrame(resul)
    display(mfg)
elif dsp == 'dft':
    # Muestra Resultado (dataframe transpuesto)
    mfg = pd.DataFrame(resul)
    display(mfg.T)

return(resul)

# -----
# Crear curva ROC
# ref: https://towardsdatascience.com/roc-curve-explained-50acab4f7bd8
# -----


def plot_roc_curve(fpr, tpr):
    plt.figure(figsize=[7, 7])
    plt.fill_between(fpr, tpr, alpha=.5)
    plt.plot([0, 1], [0, 1], linestyle='--', lw=2, label='No Skill')
    plt.plot(fpr, tpr, label="AUC=%s" % str(auc), marker='.')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.legend(loc="lower right")
    plt.title("ROC curve")
    plt.show()

# -----
# Guardar DataFrame como imagen

```

```

# ref: https://pypi.org/project/dataframe-image/
# ref: https://thispointer.com/how-to-create-a-directory-in-python/
# ----

today = date.today()
aammdd = today.strftime("%Y%m%d")

def save_df_as_png(dtf=None, n=10, filename=None, dir_name="img"):
    try:
        if not os.path.exists(dir_name):
            os.mkdir(dir_name)

        dtf.head(n).dfi.export(f'{aammdd}_{filename}.png')
        print(f"Image saved in {dir_name} directory")
    except:
        print("oops something went wrong!")

```

### 3 Importar datos

In [5]:

```

# -----
# Abrir y seleccionar archivo con GUI
# ref: https://stackoverflow.com/questions/20790926/ipython-notebook-open-select-file-with-gui-qt-dialog
# ----

%gui qt
from PyQt5.QtWidgets import QFileDialog

def gui_fname(dir=None):
    """
    Seleccione un archivo a través de un cuadro de diálogo y devuelva el nombre del archivo.
    """
    if dir is None: dir = './Datasets/'
    fname = QFileDialog.getOpenFileName(None, "Select data file...", dir, filter="All files (*);; SM Files (*.sm)")
    return fname[0]

```

In [6]:

```

# -----
-----  

VBox(children=(Button(button_style='warning', description='Seleccionar Dataset', icon='file', style=ButtonStyl...

```

```
In [7]: url = ''  
dflt_url = "Datasets/merged.csv"  
  
if out.outputs:  
    url = out.outputs[0]['text'][:-1]  
  
if len(url) <= 1 or len(out.outputs) <= 1:  
    url = dflt_url  
  
display(url)
```

'Datasets/merged.csv'

## 3.1 Datos

### 3.1.1 Carga dataset

```
In [8]: # Selección número de filas a procesar  
nrows = 3000
```

**Información:** No se esta utilizando la totalidad de la data para el análisis inicial debido a las limitaciones de hardware. La cantidad de datos modificable a través de la variable `nrows`

```
In [9]: # Carga datos  
df = pd.read_csv(url, sep='\t', na_values = np.nan, decimal=".", nrows=nrows)
```

```
In [10]: # inspection_df(df)
```

```
In [11]: # min_pairplot(df)
```

### 3.1.2 Limpieza dataset Hardware

In [12]:

```
# col_nam = {}
# for col in list(df.columns):
#     col_nam[col] = col.title().replace(" ","").replace(";", "")

# # print(col_nam)
# df.rename(columns = col_nam, inplace=True)

# # Remove special character
# for col in list(df.select_dtypes(exclude = ['float64', 'int64']).columns):
#     df[col] = df[col].str.replace(';', ' ')

# df = df.dropna()
# df = df[df.notna()]
```

### 3.1.3 Transformaciones dataset Hardware

In [13]:

```
# num_cols = list(df.columns)
# for cl in num_cols:
#     df[cl] = df[cl].apply(pd.to_numeric, downcast='integer', errors='coerce')
# #     df['Timestamp [ms]'] = df['Timestamp [ms]'].apply(pd.to_numeric, downcast='integer', errors='coerce')

# df = df.dropna()

# from datetime import datetime
# df.head()
# # df['Timestamp[Ms]'] = df['Timestamp[Ms]'].apply(datetime.fromtimestamp)
# df.iloc[:, 0] = df.iloc[:, 0].apply(datetime.fromtimestamp)
```

In [14]:

```
# df.head(10)
```

In [15]:

```
# df["Timestamp[Ms]"].to_csv(r'Datasets/fechas.txt', index=None, sep=',', mode='a')
```

## 3.2 Carga dataset transaccional

In [16]:

```

from datetime import datetime
url2 = "Datasets/datos_transaccionales.csv"

# Carga datos
df_trx = pd.read_csv(url2, sep=';', na_values = np.nan, decimal=". ", nrows=nrows)

# Elimina valores faltrantes
df_trx = df_trx.dropna()

# Cambia formato de fecha
df_trx["Fecha"] = pd.to_datetime(df_trx["Fecha"], format="%Y-%m-%d %H")
display(df_trx.head(5))

```

|   | Fecha                  | TransaccionCuenta | TransaccionResp[ms] | Servers | CPU_Int | CPuLot | Int_Car_Util | Alto_Util_Un | Alto_Util_Disco | ID | usuар | coinc | Excп  |
|---|------------------------|-------------------|---------------------|---------|---------|--------|--------------|--------------|-----------------|----|-------|-------|-------|
| 0 | 2021-03-12<br>00:05:00 | 36                | 0.08                | 2       | 0       | 2      | 0            | 6            | 156             | 3  | 488   | 2     | 18865 |
| 1 | 2021-03-12<br>00:10:00 | 12                | 0.06                | 2       | 0       | 2      | 0            | 6            | 156             | 3  | 572   | 2     | 36350 |
| 2 | 2021-03-12<br>00:15:00 | 0                 | 0.00                | 1       | 0       | 1      | 0            | 5            | 156             | 2  | 280   | 2     | 15024 |
| 3 | 2021-03-12<br>00:20:00 | 6                 | 0.00                | 1       | 0       | 1      | 0            | 5            | 148             | 2  | 300   | 2     | 23119 |
| 4 | 2021-03-12<br>00:25:00 | 17                | 0.01                | 1       | 0       | 1      | 0            | 4            | 156             | 1  | 286   | 2     | 21956 |

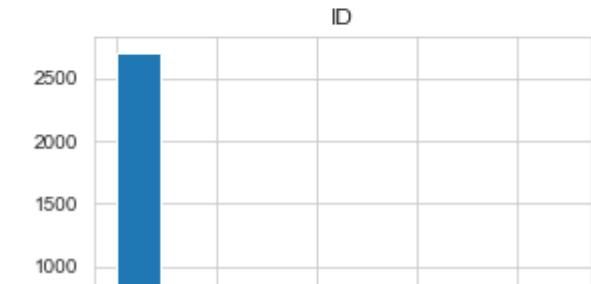
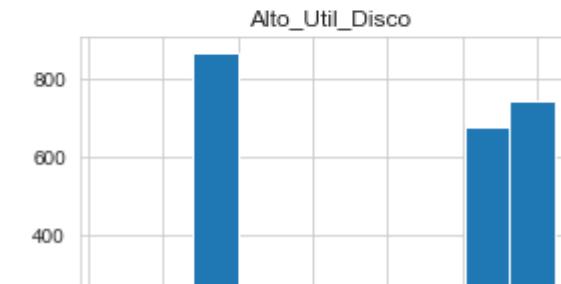
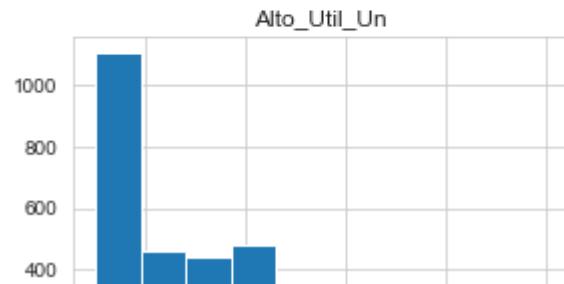
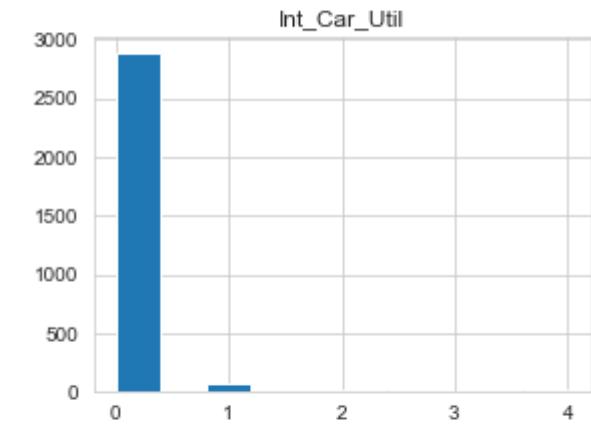
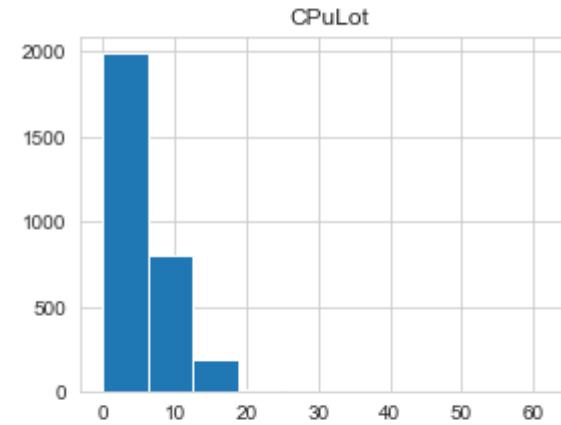
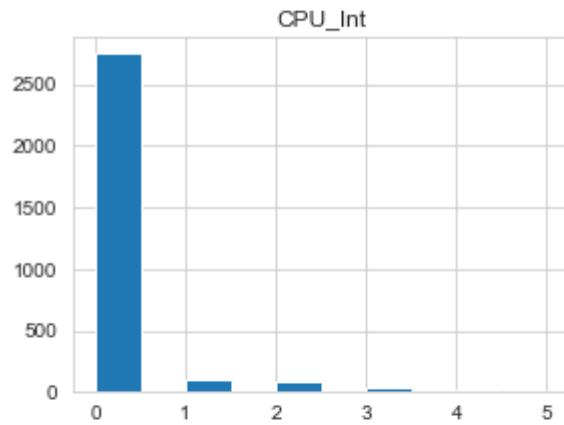
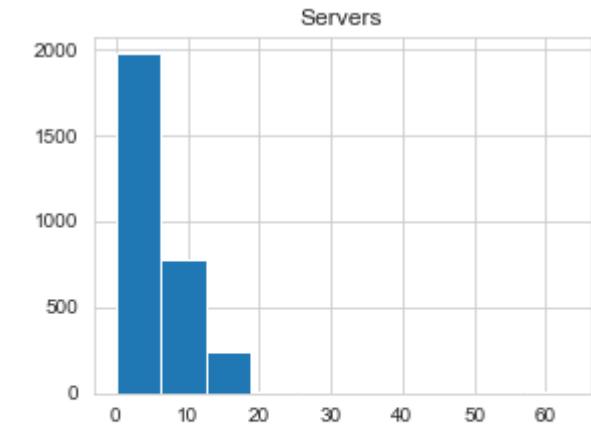
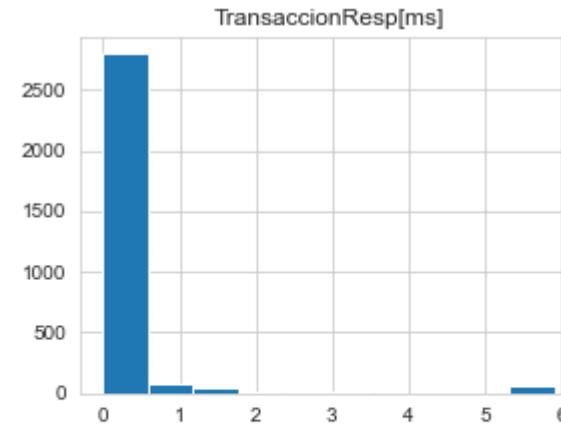
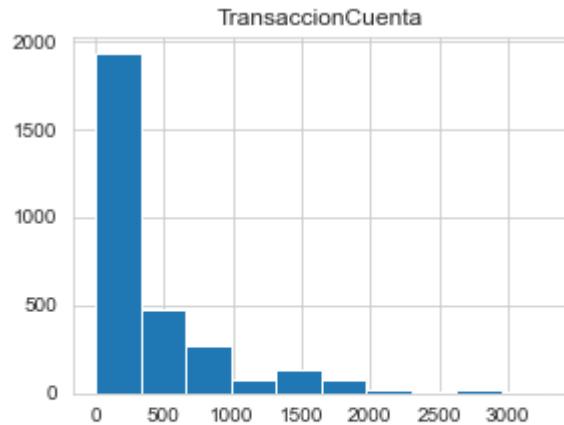
In [17]: `inspection_df(df_trx)`

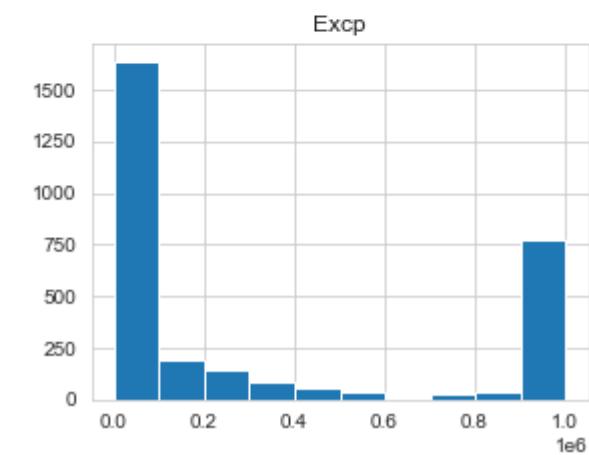
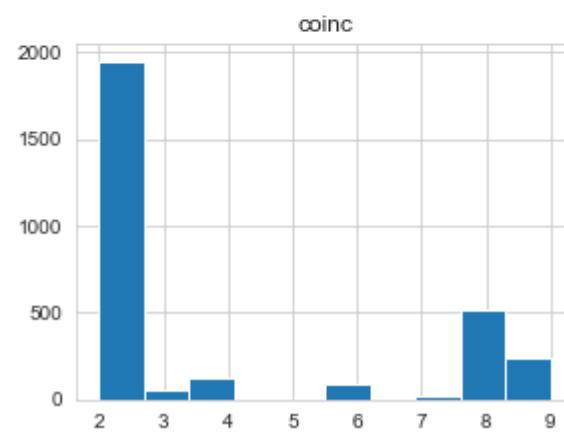
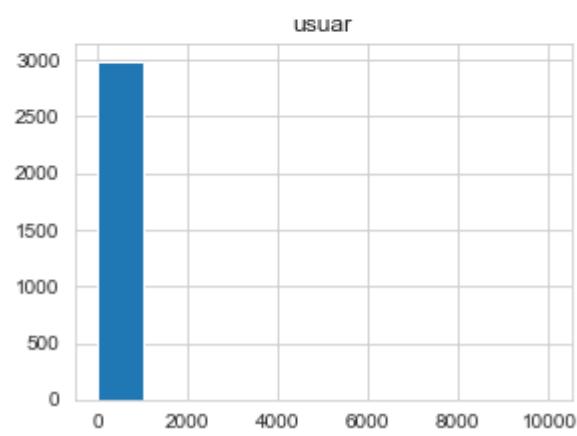
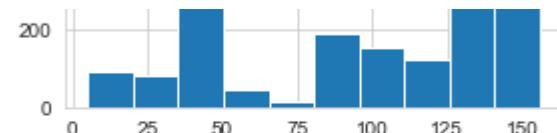
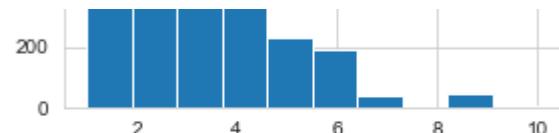
(3000, 13)

|              | Transaccion | Cuenta | Transaccion | Resp[ms] | Servers  | CPU_Int  | CPuLot   | Int_Car_Util | Alto_Util_Un | Alto_Util_Disco | ID       | usuari   | coinc    |
|--------------|-------------|--------|-------------|----------|----------|----------|----------|--------------|--------------|-----------------|----------|----------|----------|
| <b>count</b> | 3000.000    |        | 3000.000    | 3000.000 | 3000.000 | 3000.000 | 3000.000 | 3000.000     | 3000.000     | 3000.000        | 3000.000 | 3000.000 | 3000.000 |
| <b>mean</b>  | 356.811     |        | 0.202       | 5.027    | 0.168    | 4.859    | 0.075    | 2.770        | 97.364       | 0.253           | 144.448  | 3.855    | 33582    |
| <b>std</b>   | 515.984     |        | 0.833       | 5.228    | 0.649    | 5.132    | 0.447    | 1.861        | 49.843       | 1.033           | 458.112  | 2.749    | 41773    |
| <b>min</b>   | 0.000       |        | 0.000       | 0.000    | 0.000    | 0.000    | 0.000    | 1.000        | 5.000        | 0.000           | 0.000    | 2.000    | 38       |
| <b>25%</b>   | 12.000      |        | 0.000       | 1.000    | 0.000    | 1.000    | 0.000    | 1.000        | 38.000       | 0.000           | 4.000    | 2.000    | 1233     |
| <b>50%</b>   | 85.000      |        | 0.010       | 3.000    | 0.000    | 3.000    | 0.000    | 2.000        | 112.000      | 0.000           | 83.500   | 2.000    | 7659     |
| <b>75%</b>   | 554.000     |        | 0.030       | 9.000    | 0.000    | 9.000    | 0.000    | 4.000        | 140.000      | 0.000           | 214.000  | 8.000    | 97610    |
| <b>max</b>   | 3285.000    |        | 5.910       | 63.000   | 5.000    | 63.000   | 4.000    | 10.000       | 156.000      | 9.000           | 9999.000 | 9.000    | 9999.000 |

In [18]:

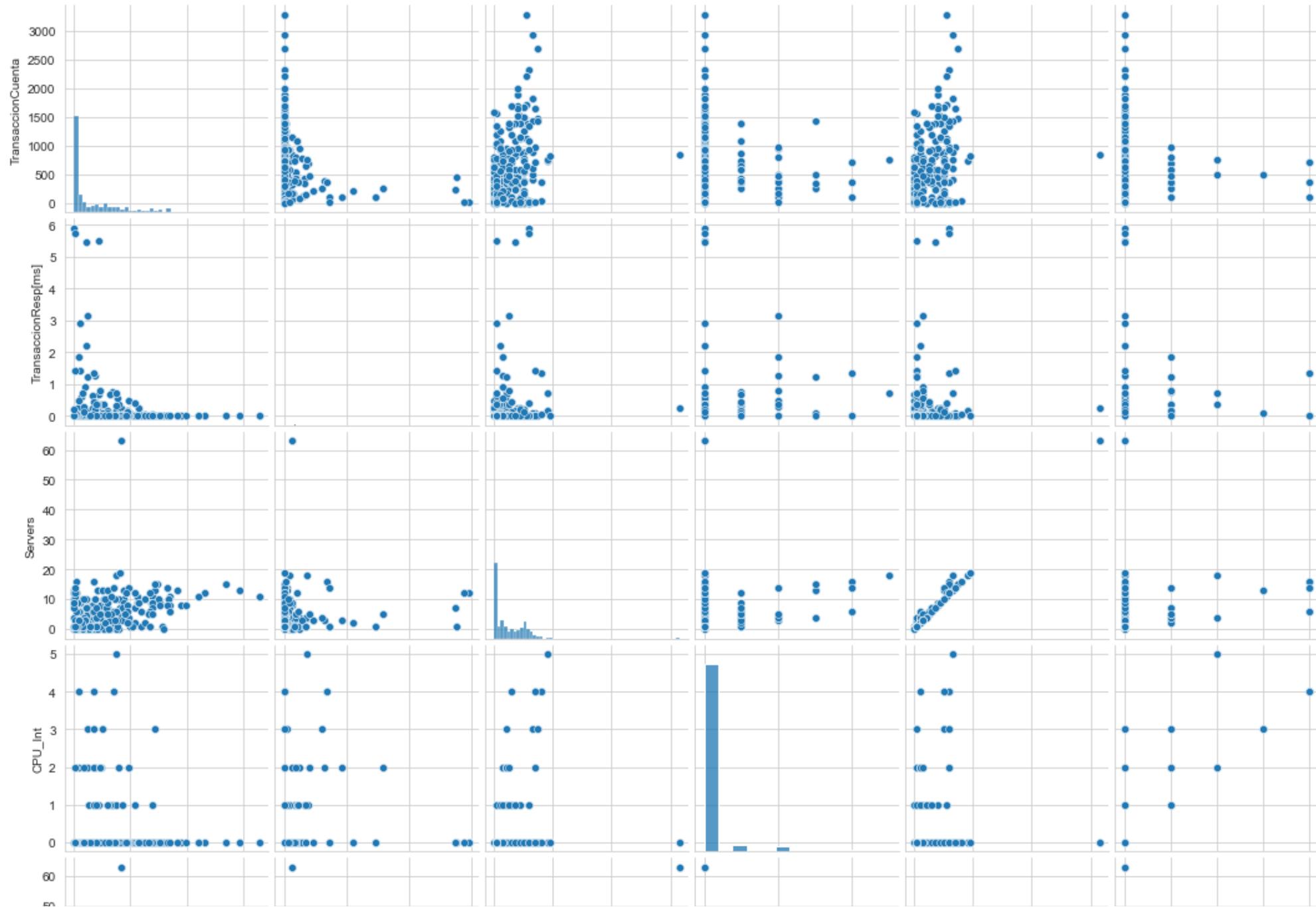
```
# -----  
# Visualizamos rápidamente las características de entrada  
# -----  
fig_dims = (16, 16)  
fig, ax = plt.subplots(figsize=fig_dims)  
df_trx.select_dtypes(exclude = ['datetime', 'object']).hist(ax=ax)  
plt.show()
```

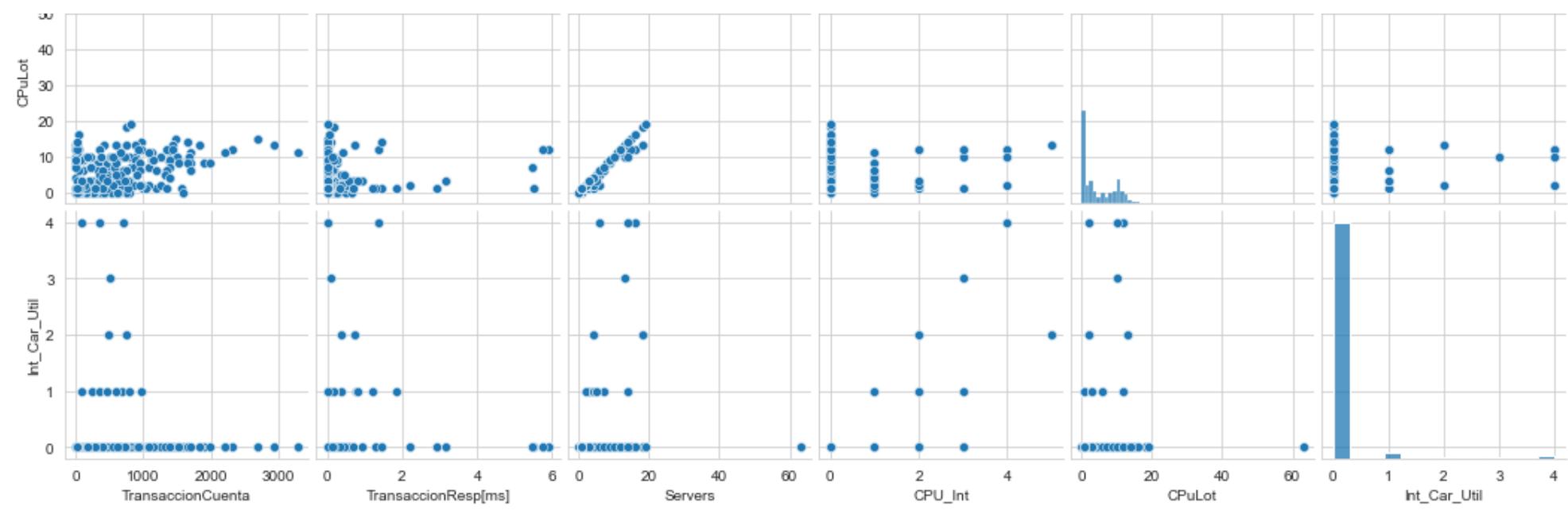




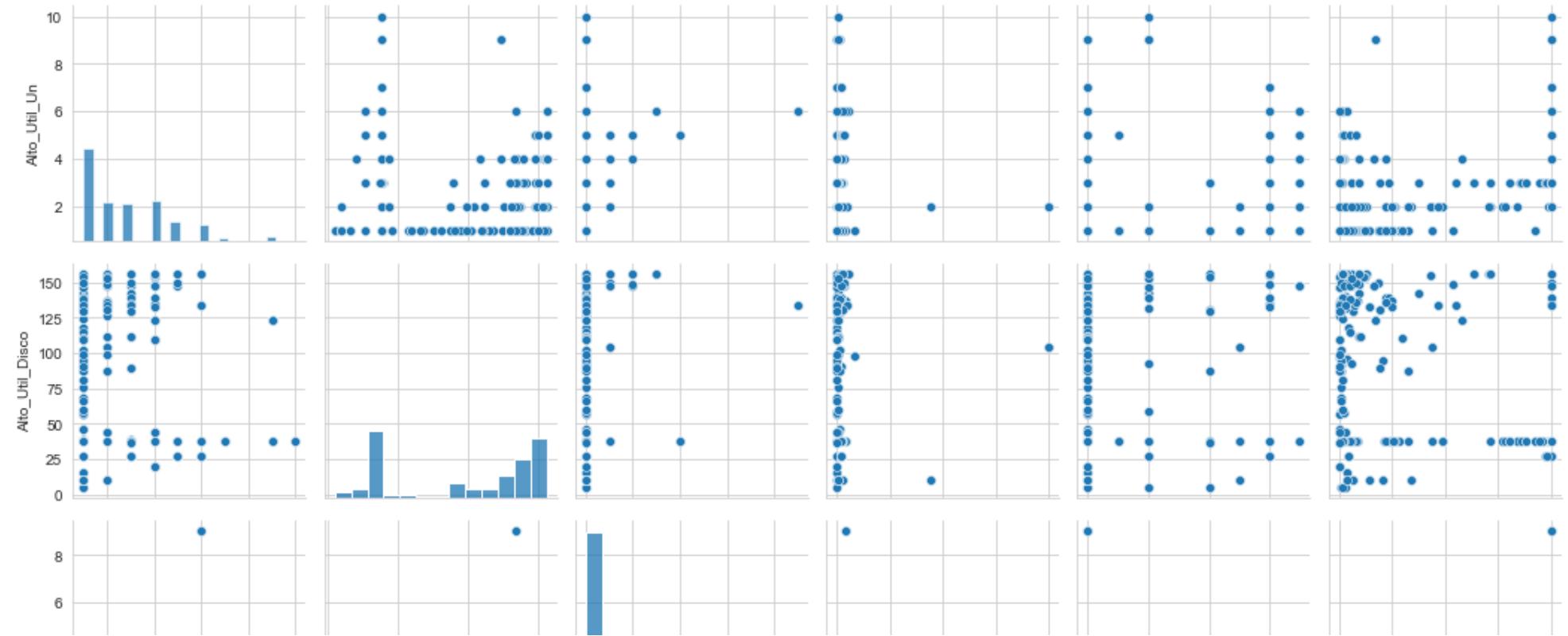
In [19]: min\_pairplot(df\_trx)

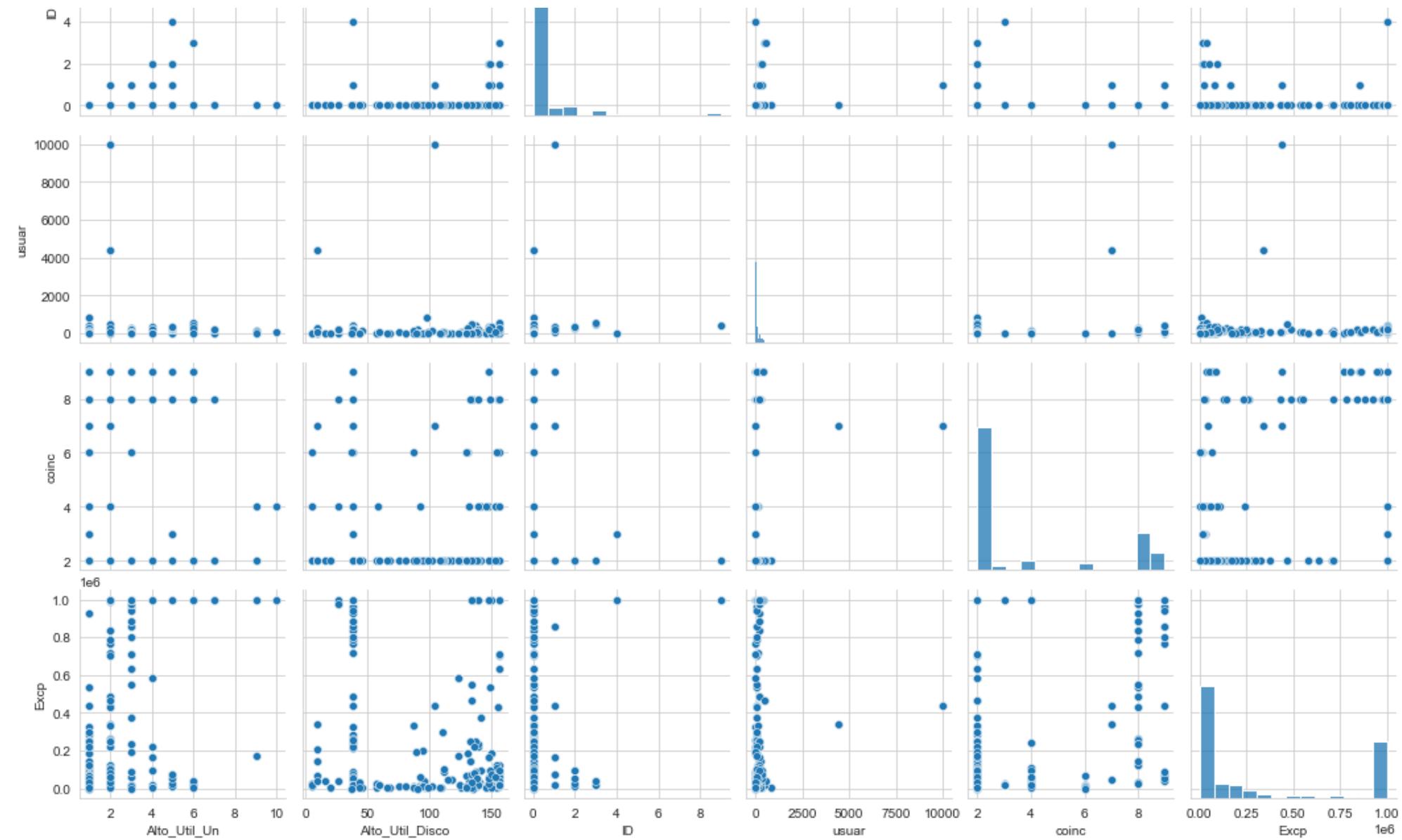
```
['TransaccionCuenta', 'TransaccionResp[ms]', 'Servers', 'CPU_Int', 'CPuLot', 'Int_Car_Util']
```



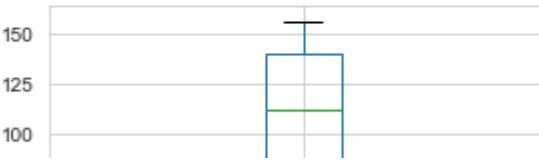
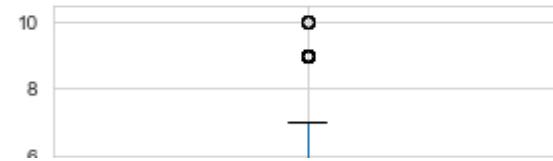
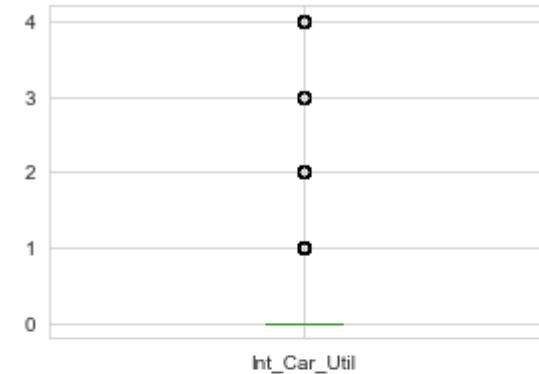
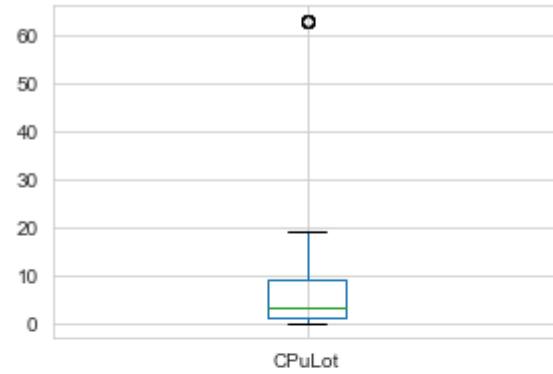
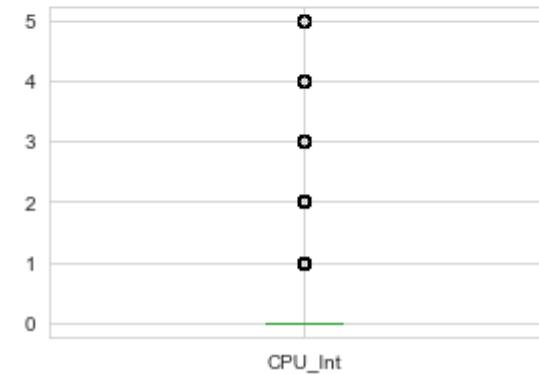
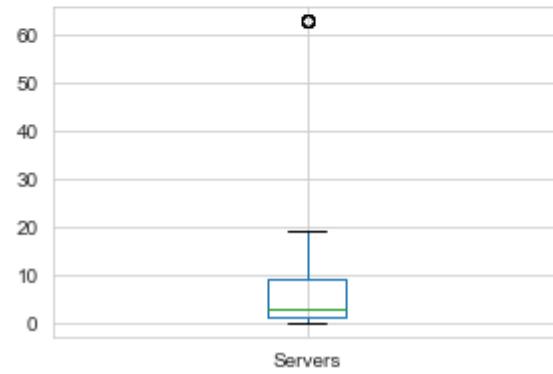
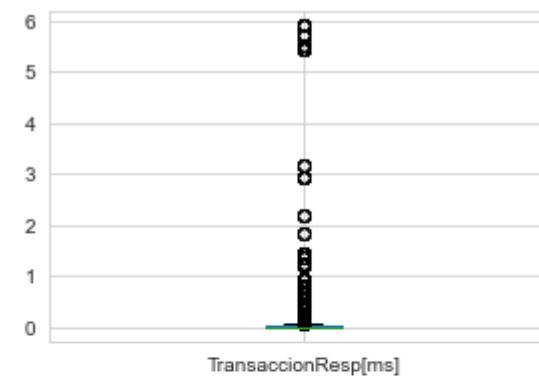


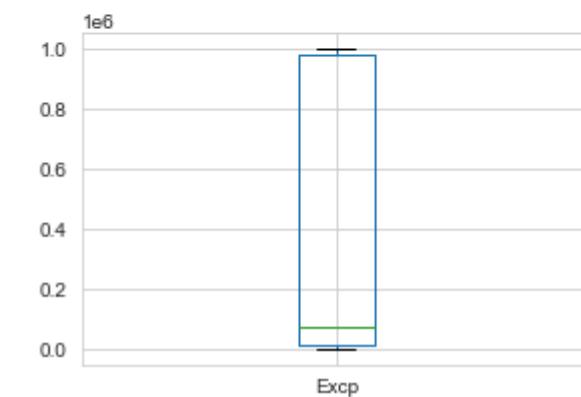
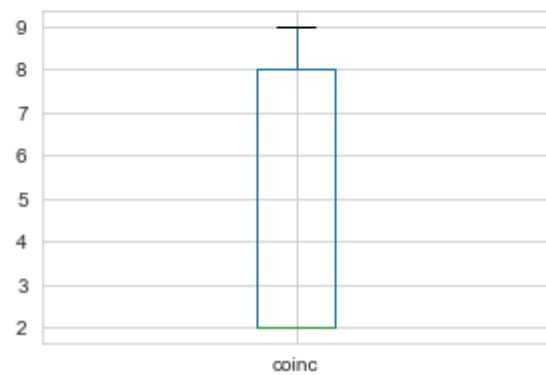
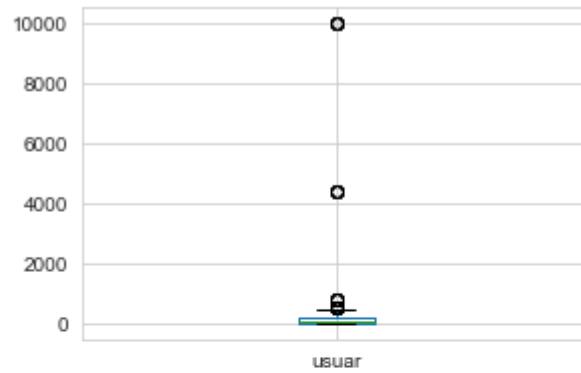
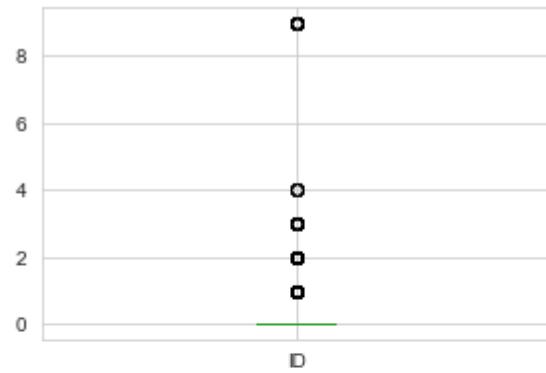
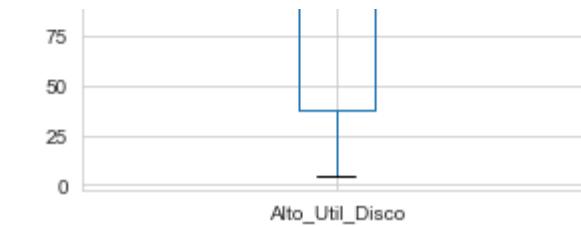
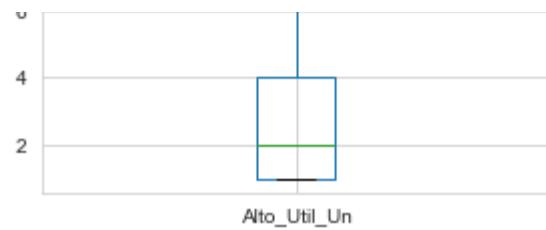
```
['Alto_Util_Un', 'Alto_Util_Disco', 'ID', 'usuar', 'coinc', 'Excp']
```





```
In [20]: df_trx.select_dtypes(exclude = ['datetime', 'object']).plot(kind='box', subplots=True, layout=(7,2), figsize=(10,25), sharex=False)
```

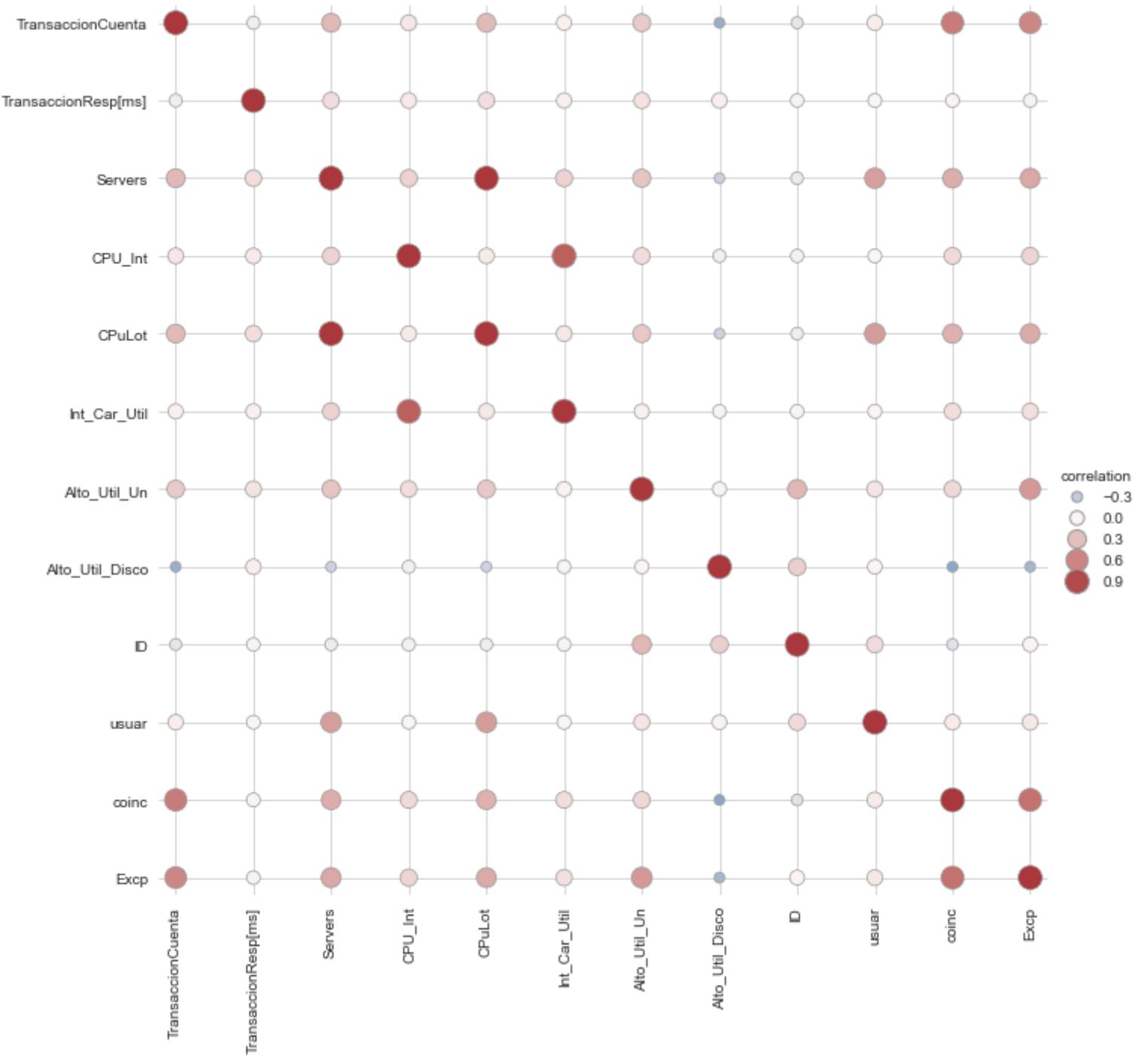




In [21]:

```
corr_mat = df_trx.corr().stack().reset_index(name="correlation")

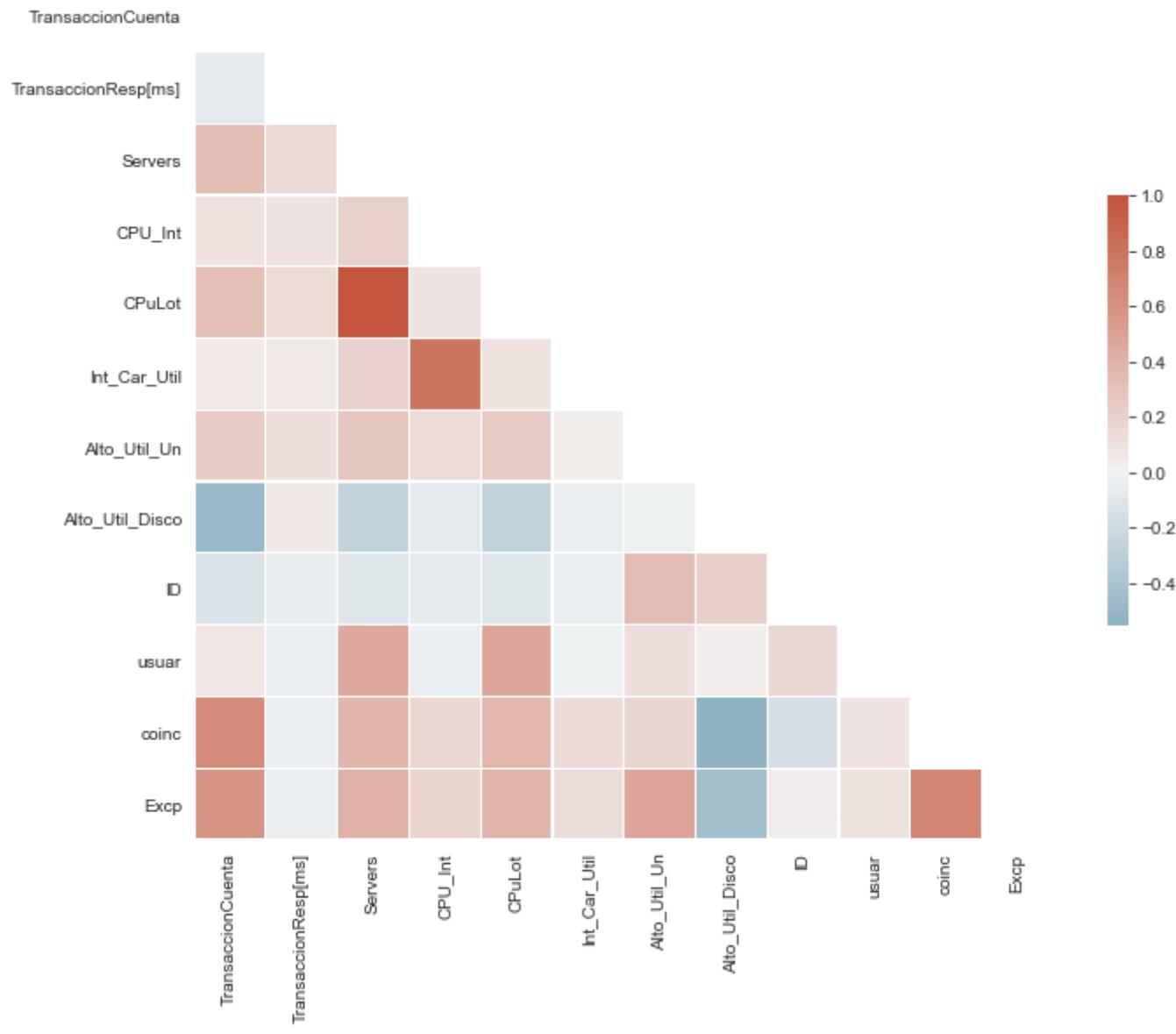
g = sns.relplot(
    data=corr_mat,
    x="level_0", y="level_1", hue="correlation", size="correlation",
    palette="vlag", hue_norm=(-1, 1), edgecolor=".6",
    height=10, sizes=(50, 250), size_norm=(-.2, .8),
)
g.set(xlabel="", ylabel="", aspect="equal")
g.despine(left=True, bottom=True)
g.ax.margins(.02)
for label in g.ax.get_xticklabels():
    label.set_rotation(90)
for artist in g.legend.legendHandles:
    artist.set_edgecolor(".6")
```



In [22]:

```
# -----  
# Matriz de correlación diagonal  
# ref: https://seaborn.pydata.org/examples/many\_pairwise\_correlations.html?highlight=corr  
# -----  
  
corr = df_trx.select_dtypes(exclude = ['datetime', 'object']).corr()  
mask = np.triu(np.ones_like(corr, dtype=bool))  
f, ax = plt.subplots(figsize=(11, 9))  
cmap = sns.diverging_palette(230, 20, as_cmap=True)  
sns.heatmap(corr,  
            mask=mask,  
            cmap=cmap,  
            vmax=1,  
            center=0,  
            square=True,  
            linewidths=.3,  
            cbar_kws={"shrink": .5}  
        )
```

Out[22]: <AxesSubplot:>



### 3.3 Carga dataset Fuente de datos de hardware

In [23]:

```

url3 = "Datasets/datos_hardware_clai.csv"

# Carga datos
df_hrd = pd.read_csv(url3, sep=';', na_values = np.nan, decimal=". ", nrows=nrows)
display(df_hrd.head(5))

#Se eliminan algunas columnas
no_utiles = ['tiempo']

df_hrd = delete_columns(_df=df_hrd, del_cols=no_utiles)
df_hrd.head(50).dfi.export(f'img/{aammdd}_df_hrd_clai.png')
df_hrd.head(5)

```

|   | tiempo | Promedio_ES | Promedio_Lecturas | Promedio_Grabaciones | Promedio_K_ES | Promedio_uti | Util_alta | Uni_util_alta | Tiempo_Service_Alto | Unit_Service_al |
|---|--------|-------------|-------------------|----------------------|---------------|--------------|-----------|---------------|---------------------|-----------------|
| 0 | 02:15  | 992.2       | 639.2             | 353.0                | 8.6           | 32.8         | 71.0      | 7             | 0.003               |                 |
| 1 | 02:20  | 987.0       | 524.0             | 462.9                | 11.0          | 37.2         | 81.3      | 7             | 0.008               |                 |
| 2 | 02:25  | 959.4       | 730.0             | 229.3                | 4.5           | 34.0         | 73.5      | 7             | 0.004               |                 |
| 3 | 02:30  | 947.2       | 717.4             | 229.8                | 4.5           | 33.2         | 72.1      | 7             | 0.004               |                 |
| 4 | 02:35  | 954.2       | 721.3             | 232.8                | 4.4           | 33.0         | 73.1      | 7             | 0.004               |                 |

Se eliminaron satisfactoriamente las columnas del dataset.

Las siguientes campos no existen como columnas en el dataset:

-----  
['tiempo']

Out[23]:

|   | Promedio_ES | Promedio_Lecturas | Promedio_Grabaciones | Promedio_K_ES | Promedio_uti | Util_alta | Uni_util_alta | Tiempo_Service_Alto | Unit_Service_alta | Espa |
|---|-------------|-------------------|----------------------|---------------|--------------|-----------|---------------|---------------------|-------------------|------|
| 0 | 992.2       | 639.2             | 353.0                | 8.6           | 32.8         | 71.0      | 7             | 0.003               | 7                 |      |
| 1 | 987.0       | 524.0             | 462.9                | 11.0          | 37.2         | 81.3      | 7             | 0.008               | 7                 |      |
| 2 | 959.4       | 730.0             | 229.3                | 4.5           | 34.0         | 73.5      | 7             | 0.004               | 7                 |      |
| 3 | 947.2       | 717.4             | 229.8                | 4.5           | 33.2         | 72.1      | 7             | 0.004               | 7                 |      |
| 4 | 954.2       | 721.3             | 232.8                | 4.4           | 33.0         | 73.1      | 7             | 0.004               | 7                 |      |

In [24]:

```
df_hrd.describe().dfi.export(f'img/{aammdd}_describe_hrd_clai.png')
inspection_df(df_hrd)
```

(3000, 10)

|       | Promedio_ES | Promedio_Lecturas | Promedio_Grabaciones | Promedio_K_ES | Promedio_utí | Util_alta | Uni_util_alta | Tiempo_Service_Alto | Unit_Service_alta |
|-------|-------------|-------------------|----------------------|---------------|--------------|-----------|---------------|---------------------|-------------------|
| count | 3000.000    | 3000.000          | 3000.000             | 3000.000      | 3000.000     | 3000.000  | 3000.000      | 3.000e+03           | 3000.000          |
| mean  | 26.411      | 17.428            | 8.937                | 8.667         | 0.609        | 1.223     | 6.858         | 7.813e-04           | 4.099             |
| std   | 116.803     | 98.769            | 24.648               | 5.120         | 3.743        | 7.748     | 0.664         | 8.925e-04           | 2.246             |
| min   | 1.000       | 0.000             | 0.900                | 4.400         | 0.000        | 0.000     | 1.000         | 0.000e+00           | 1.000             |
| 25%   | 2.800       | 0.100             | 2.300                | 6.100         | 0.000        | 0.000     | 7.000         | 3.000e-04           | 2.000             |
| 50%   | 5.100       | 0.200             | 4.300                | 7.700         | 0.000        | 0.000     | 7.000         | 5.000e-04           | 4.000             |
| 75%   | 9.400       | 1.100             | 8.400                | 10.000        | 0.000        | 0.000     | 7.000         | 9.000e-04           | 6.000             |
| max   | 992.200     | 743.300           | 462.900              | 53.700        | 37.200       | 81.300    | 7.000         | 8.200e-03           | 7.000             |

```
['Promedio_ES',
 'Promedio_Lecturas',
 'Promedio_Grabaciones',
 'Promedio_K_ES',
 'Promedio_utí',
 'Util_alta',
 'Uni_util_alta',
 'Tiempo_Service_Alto',
 'Unit_Service_alta',
 'Espacio_Disco_Utilizado']
```

Promedio\_ES

float64

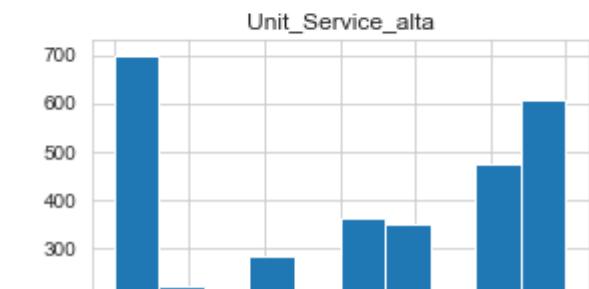
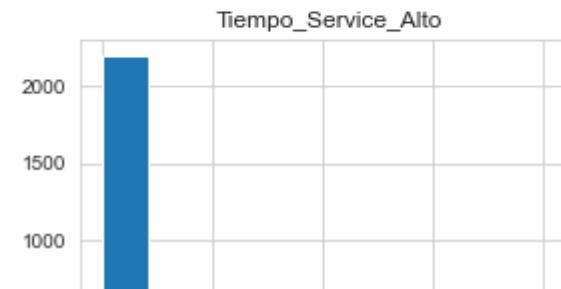
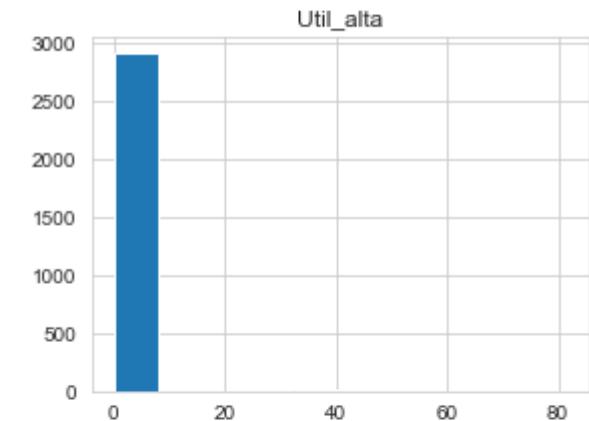
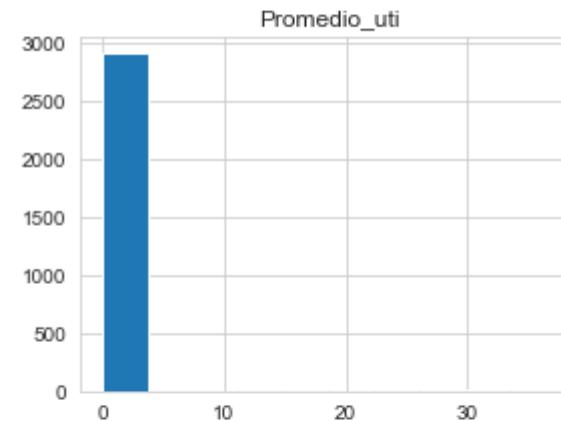
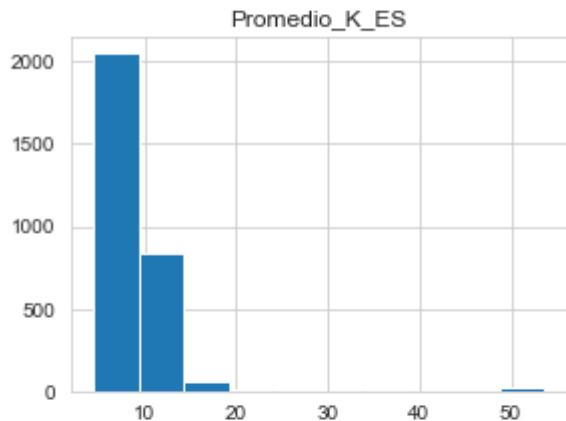
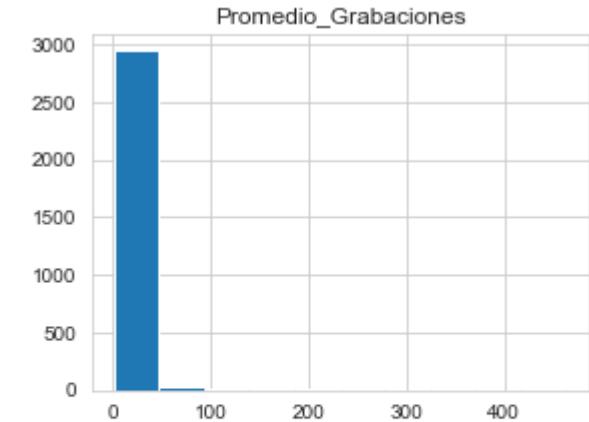
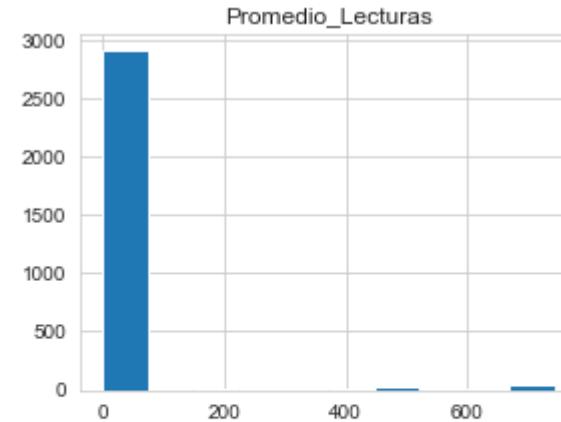
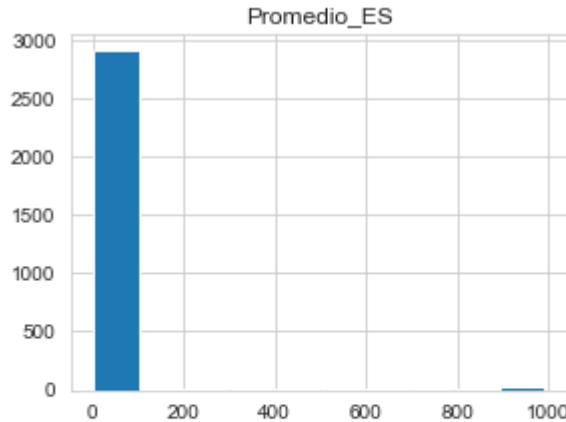
```
Promedio_Lecturas      float64
Promedio_Grabaciones   float64
Promedio_K_ES          float64
Promedio_utí           float64
Util_alta              float64
Uni_utí_alta           int64
Tiempo_Service_Alto   float64
Unit_Service_alta      int64
Espacio_Disco_Utilizado float64
dtype: object
```

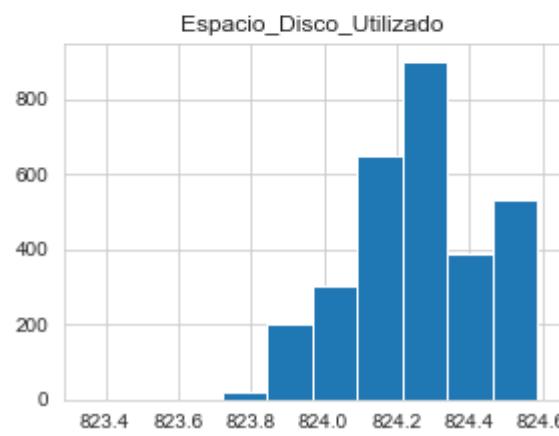
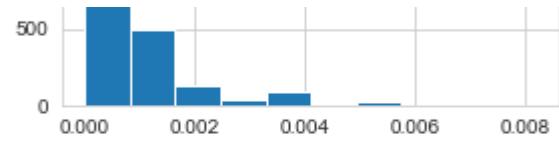
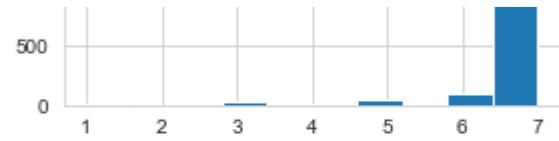
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
 --- 
 0   Promedio_ES        3000 non-null    float64
 1   Promedio_Lecturas  3000 non-null    float64
 2   Promedio_Grabaciones 3000 non-null    float64
 3   Promedio_K_ES      3000 non-null    float64
 4   Promedio_utí       3000 non-null    float64
 5   Util_alta          3000 non-null    float64
 6   Uni_utí_alta       3000 non-null    int64  
 7   Tiempo_Service_Alto 3000 non-null    float64
 8   Unit_Service_alta  3000 non-null    int64  
 9   Espacio_Disco_Utilizado 3000 non-null    float64
dtypes: float64(8), int64(2)
memory usage: 234.5 KB
```

```
None
```

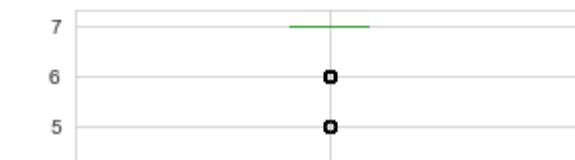
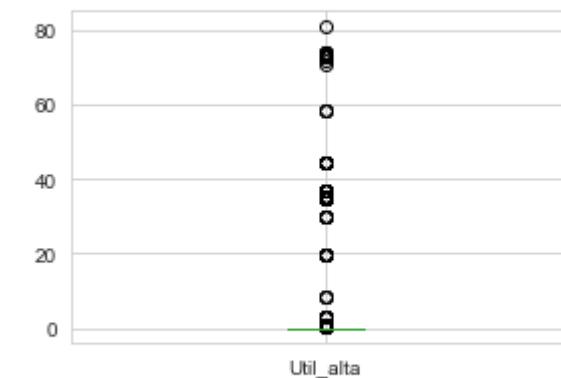
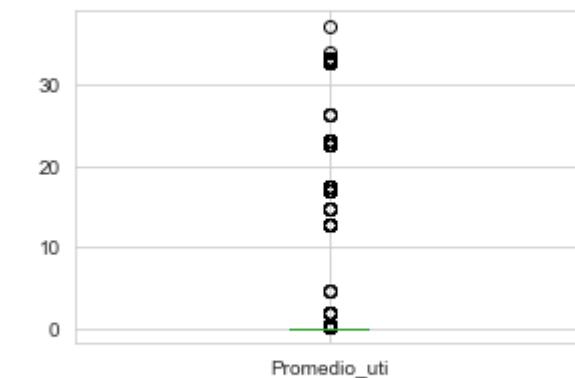
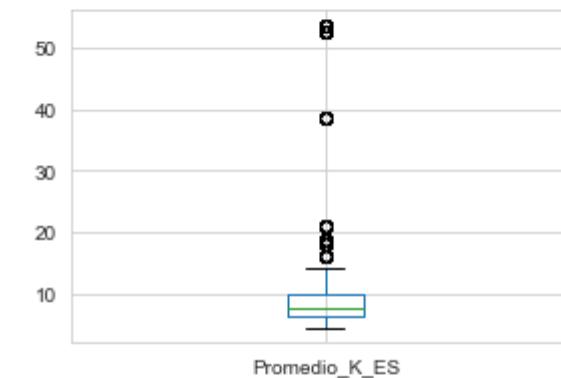
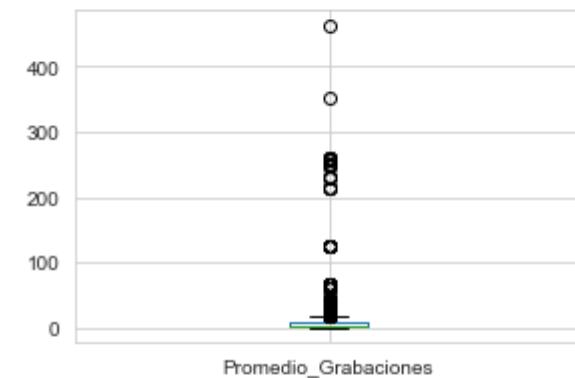
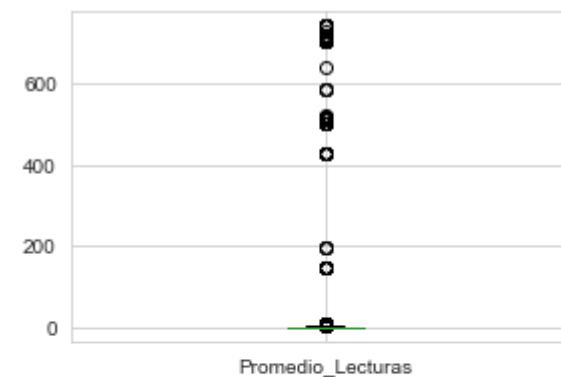
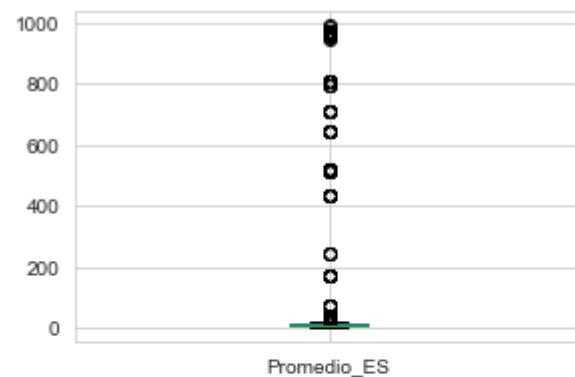
In [25]:

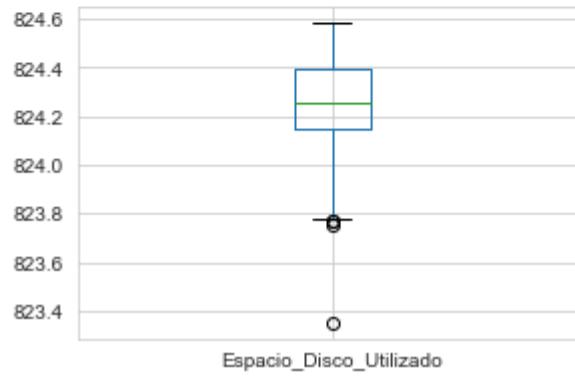
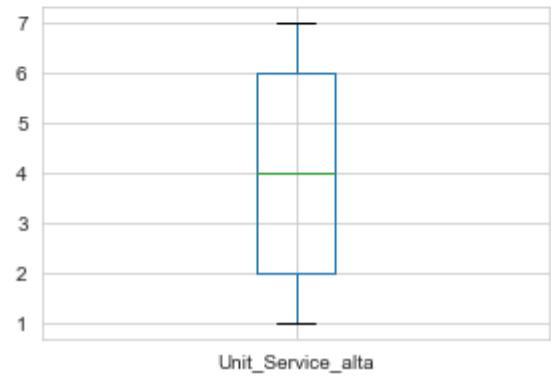
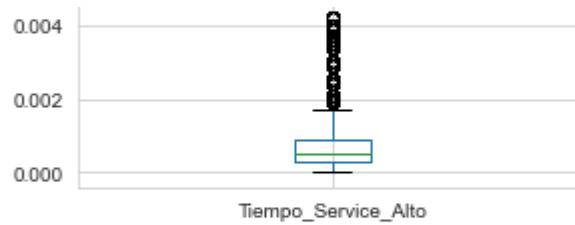
```
# -----  
# Visualizamos rápidamente las características de entrada  
# -----  
fig_dims = (16, 16)  
fig, ax = plt.subplots(figsize=fig_dims)  
df_hrd.select_dtypes(exclude = ['datetime', 'object']).hist(ax=ax)  
plt.show()
```





```
In [26]: df_hrd.select_dtypes(exclude = ['datetime', 'object']).plot(kind='box', subplots=True, layout=(7,2), figsize=(10,25), sharex=False)
```

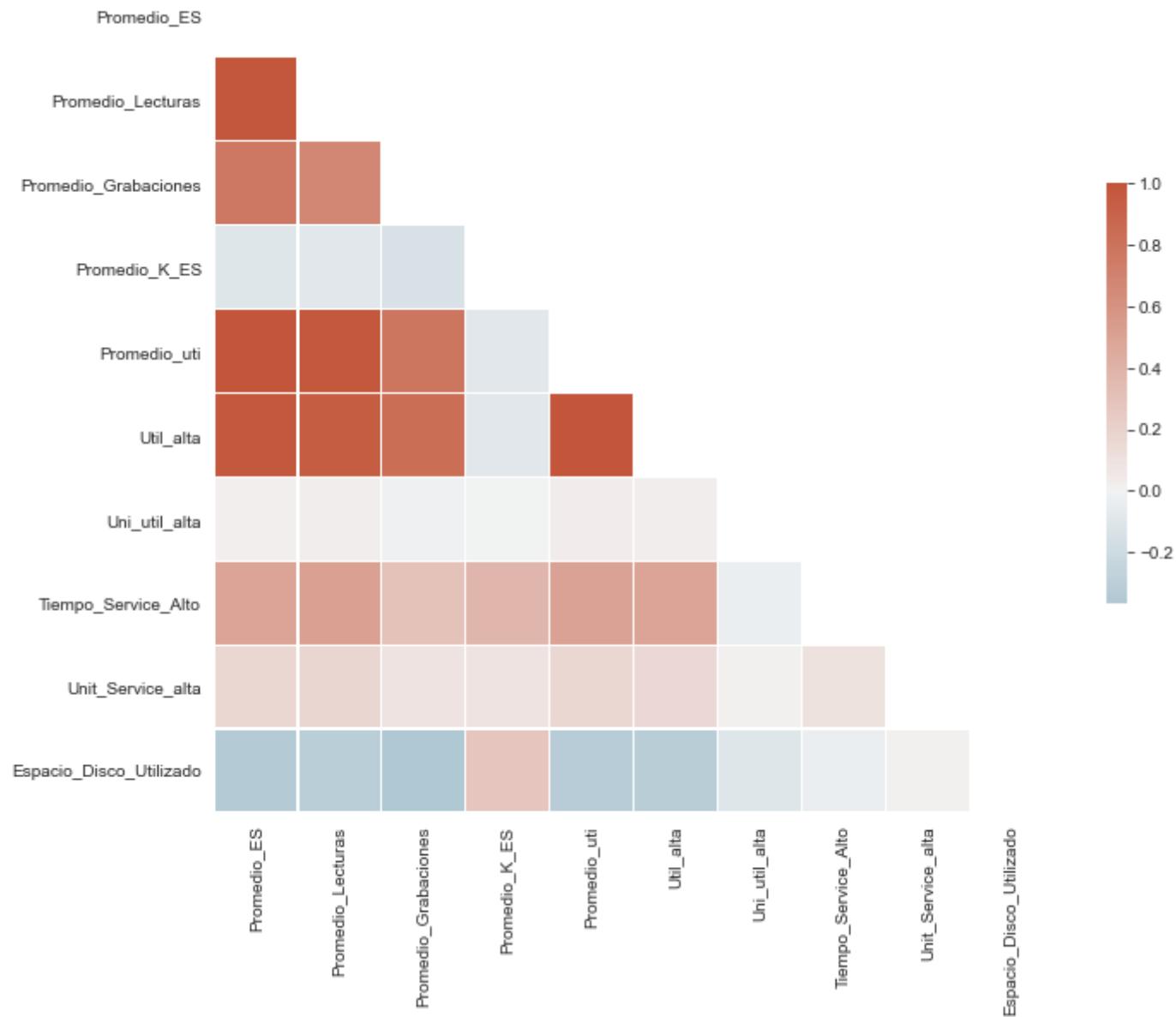




In [27]:

```
# -----  
# Matriz de correlación diagonal  
# ref: https://seaborn.pydata.org/examples/many\_pairwise\_correlations.html?highlight=corr  
# -----  
corr = df_hrd.select_dtypes(exclude = ['datetime', 'object']).corr()  
mask = np.triu(np.ones_like(corr, dtype=bool))  
f, ax = plt.subplots(figsize=(11, 9))  
cmap = sns.diverging_palette(230, 20, as_cmap=True)  
sns.heatmap(corr,  
            mask=mask,  
            cmap=cmap,  
            vmax=1,  
            center=0,  
            square=True,  
            linewidths=.3,  
            cbar_kws={"shrink": .5}  
)
```

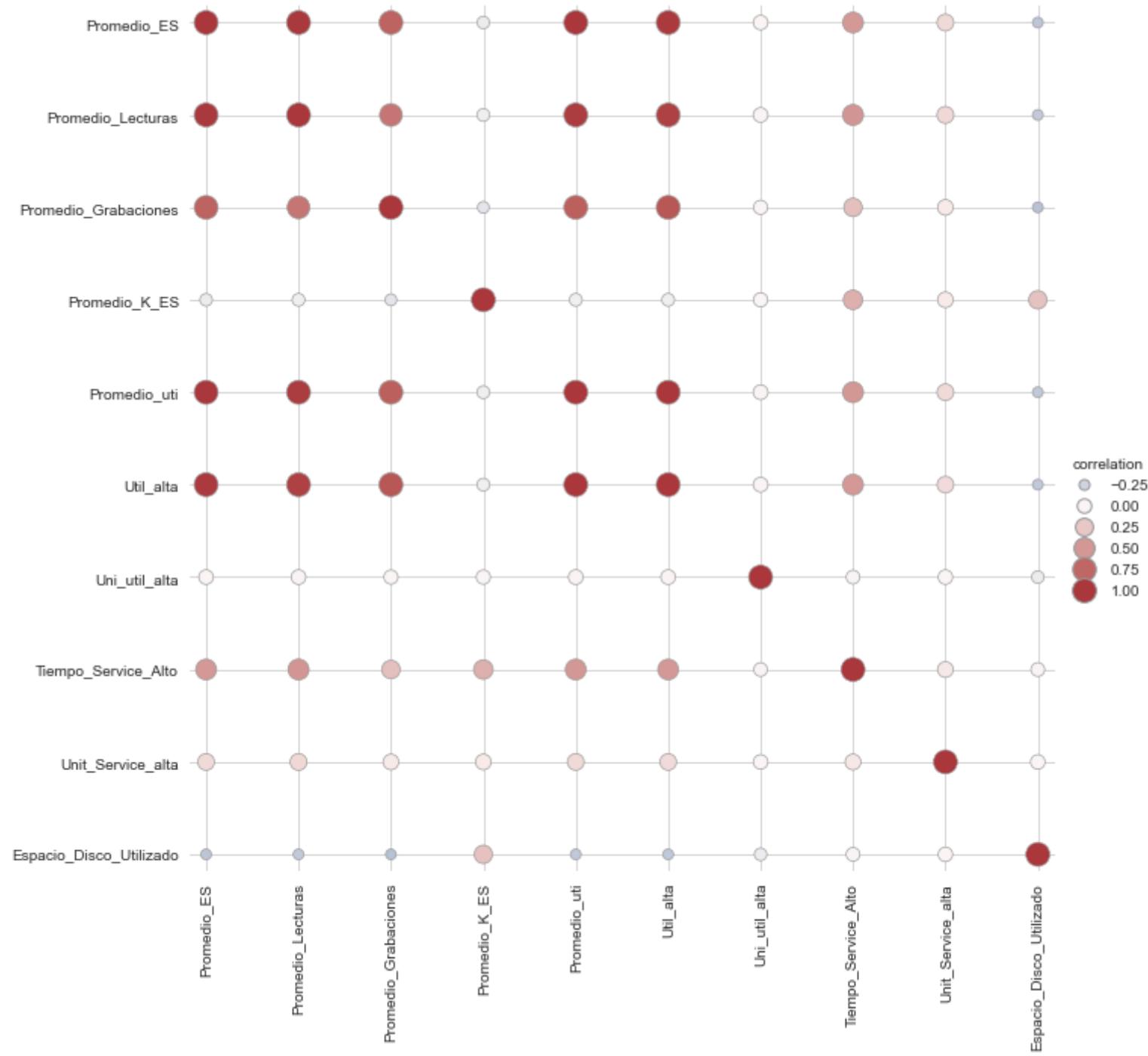
Out[27]: <AxesSubplot:>



In [28]:

```
corr_mat = df_hrd.corr().stack().reset_index(name="correlation")

g = sns.relplot(
    data=corr_mat,
    x="level_0", y="level_1", hue="correlation", size="correlation",
    palette="vlag", hue_norm=(-1, 1), edgecolor=".6",
    height=10, sizes=(50, 250), size_norm=(-.2, .8),
)
g.set(xlabel="", ylabel="", aspect="equal")
g.despine(left=True, bottom=True)
g.ax.margins(.02)
for label in g.ax.get_xticklabels():
    label.set_rotation(90)
for artist in g.legend.legendHandles:
    artist.set_edgecolor(".6")
```



### 3.3.1 Transformaciones dataset Trx

In [29]:

```
print(f'Hay {df_trx.isnull().any().sum()} columnas en el conjunto de datos con datos faltantes.')
missing_values_count = df_trx.isnull().sum()
missing_values_count = missing_values_count[missing_values_count.values > 0]
Cls2drop = dict(missing_values_count)
print(Cls2drop)
```

Hay 0 columnas en el conjunto de datos con datos faltantes.

{}

In [30]:

```
print(f'Hay {df_hrd.isnull().any().sum()} columnas en el conjunto de datos con datos faltantes.')
missing_values_count = df_hrd.isnull().sum()
missing_values_count = missing_values_count[missing_values_count.values > 0]
Cls2drop = dict(missing_values_count)
print(Cls2drop)
```

Hay 0 columnas en el conjunto de datos con datos faltantes.

{}

In [31]:

```
print("df_trx: ", check_unique(df_trx))
print("df_hrd: ", check_unique(df_hrd))
```

df\_trx: ['DataFrame no tiene columnas con valores únicos']

df\_hrd: ['DataFrame no tiene columnas con valores únicos']

In [32]:

```
print(df_hrd.info(verbose=False, memory_usage="deep"), "\n")
```

```
print(df_trx.info(verbose=False, memory_usage="deep"))
```

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 3000 entries, 0 to 2999

Columns: 10 entries, Promedio\_ES to Espacio\_Disco\_Utilizado

dtypes: float64(8), int64(2)

memory usage: 234.5 KB

None

<class 'pandas.core.frame.DataFrame'>

Int64Index: 3000 entries, 0 to 2999

Columns: 13 entries, Fecha to Excp

dtypes: datetime64[ns](1), float64(1), int64(11)

memory usage: 328.1 KB

None

## 4 Transformaciones

### 4.1 Transformaciones dataset Transaccional

```
In [33]: df_clai = pd.merge(df_trx, df_hrd, left_index=True, right_index=True)

df_clai.head(50).dfi.export(f'img/{aammdd}_df_trx_clai.png')
display(df_clai.head(5))
print(df_clai.info(verbose=False, memory_usage="deep"))
```

|   | Fecha               | TransaccionCuenta | TransaccionResp[ms] | Servers | CPU_Int | CPuLot | Int_Car_Util | Alto_Util_Un | Alto_Util_Disco | ID | usuар | coinc | Excp  | Promedio_ |
|---|---------------------|-------------------|---------------------|---------|---------|--------|--------------|--------------|-----------------|----|-------|-------|-------|-----------|
| 0 | 2021-03-12 00:05:00 | 36                | 0.08                | 2       | 0       | 2      | 0            | 6            | 156             | 3  | 488   | 2     | 18865 | 99        |
| 1 | 2021-03-12 00:10:00 | 12                | 0.06                | 2       | 0       | 2      | 0            | 6            | 156             | 3  | 572   | 2     | 36350 | 98        |
| 2 | 2021-03-12 00:15:00 | 0                 | 0.00                | 1       | 0       | 1      | 0            | 5            | 156             | 2  | 280   | 2     | 15024 | 95        |
| 3 | 2021-03-12 00:20:00 | 6                 | 0.00                | 1       | 0       | 1      | 0            | 5            | 148             | 2  | 300   | 2     | 23119 | 94        |
| 4 | 2021-03-12 00:25:00 | 17                | 0.01                | 1       | 0       | 1      | 0            | 4            | 156             | 1  | 286   | 2     | 21956 | 95        |

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3000 entries, 0 to 2999
Columns: 23 entries, Fecha to Espacio_Disco_Utilizado
dtypes: datetime64[ns](1), float64(9), int64(13)
memory usage: 642.5 KB
None
```

### 4.2 Se eliminan Columnas

In [34]: df\_clai

Out[34]:

|      | Fecha               | TransaccionCuenta | TransaccionResp[ms] | Servers | CPU_Int | CPuLot | Int_Car_Util | Alto_Util_Un | Alto_Util_Disco | ID  | usuari | coinc | Excp | Promes |
|------|---------------------|-------------------|---------------------|---------|---------|--------|--------------|--------------|-----------------|-----|--------|-------|------|--------|
| 0    | 2021-03-12 00:05:00 | 36                | 0.08                | 2       | 0       | 2      | 0            | 6            |                 | 156 | 3      | 488   | 2    | 18865  |
| 1    | 2021-03-12 00:10:00 | 12                | 0.06                | 2       | 0       | 2      | 0            | 6            |                 | 156 | 3      | 572   | 2    | 36350  |
| 2    | 2021-03-12 00:15:00 | 0                 | 0.00                | 1       | 0       | 1      | 0            | 5            |                 | 156 | 2      | 280   | 2    | 15024  |
| 3    | 2021-03-12 00:20:00 | 6                 | 0.00                | 1       | 0       | 1      | 0            | 5            |                 | 148 | 2      | 300   | 2    | 23119  |
| 4    | 2021-03-12 00:25:00 | 17                | 0.01                | 1       | 0       | 1      | 0            | 4            |                 | 156 | 1      | 286   | 2    | 21956  |
| ...  | ...                 | ...               | ...                 | ...     | ...     | ...    | ...          | ...          | ...             | ... | ...    | ...   | ...  | ...    |
| 2995 | 2021-03-22 09:40:15 | 6                 | 0.19                | 9       | 0       | 9      | 0            | 4            |                 | 109 | 0      | 1     | 2    | 2089   |
| 2996 | 2021-03-22 09:45:15 | 168               | 0.15                | 10      | 0       | 10     | 0            | 4            |                 | 123 | 0      | 12    | 2    | 582674 |
| 2997 | 2021-03-22 09:50:15 | 4                 | 5.91                | 12      | 0       | 12     | 0            | 1            |                 | 134 | 0      | 11    | 2    | 32399  |
| 2998 | 2021-03-22 09:55:15 | 12                | 5.73                | 12      | 0       | 12     | 0            | 9            |                 | 123 | 0      | 22    | 2    | 170760 |
| 2999 | 2021-03-22 10:00:15 | 14                | 1.44                | 14      | 0       | 14     | 0            | 1            |                 | 130 | 0      | 0     | 2    | 5920   |

3000 rows × 23 columns

In [35]:

```
# https://pypi.org/project/dataframe-image/
import dataframe_image as dfi
df_clai.head(5).dfi.export('df_clai.png')

df_clai.head(5)
```

```
# cm = sns.light_palette("seagreen", as_cmap=True)
# styled_table = df_clai.style.background_gradient(cmap=cm)
# styled_table
```

Out[35]:

|   | Fecha               | TransaccionCuenta | TransaccionResp[ms] | Servers | CPU_Int | CPuLot | Int_Car_Util | Alto_Util_Un | Alto_Util_Disco | ID | usuар | coinc | Excp  | Promedio_ |
|---|---------------------|-------------------|---------------------|---------|---------|--------|--------------|--------------|-----------------|----|-------|-------|-------|-----------|
| 0 | 2021-03-12 00:05:00 | 36                | 0.08                | 2       | 0       | 2      | 0            | 6            | 156             | 3  | 488   | 2     | 18865 | 99        |
| 1 | 2021-03-12 00:10:00 | 12                | 0.06                | 2       | 0       | 2      | 0            | 6            | 156             | 3  | 572   | 2     | 36350 | 98        |
| 2 | 2021-03-12 00:15:00 | 0                 | 0.00                | 1       | 0       | 1      | 0            | 5            | 156             | 2  | 280   | 2     | 15024 | 95        |
| 3 | 2021-03-12 00:20:00 | 6                 | 0.00                | 1       | 0       | 1      | 0            | 5            | 148             | 2  | 300   | 2     | 23119 | 94        |
| 4 | 2021-03-12 00:25:00 | 17                | 0.01                | 1       | 0       | 1      | 0            | 4            | 156             | 1  | 286   | 2     | 21956 | 95        |



In [36]:

```
display(df_clai.describe().round(2))
df_clai.describe().round(2).dfi.export(f'img/{aammdd}_describe_trx_clai.png')
df_clai = clean_outliers_zscore(df_clai, show_outliers=False)
```

|              | TransaccionCuenta | TransaccionResp[ms] | Servers | CPU_Int | CPuLot  | Int_Car_Util | Alto_Util_Un | Alto_Util_Disco | ID      | usuар   | coinc   | Excp      | Pr      |
|--------------|-------------------|---------------------|---------|---------|---------|--------------|--------------|-----------------|---------|---------|---------|-----------|---------|
| <b>count</b> | 3000.00           | 3000.00             | 3000.00 | 3000.00 | 3000.00 | 3000.00      | 3000.00      | 3000.00         | 3000.00 | 3000.00 | 3000.00 | 3000.00   | 3000.00 |
| <b>mean</b>  | 356.81            | 0.20                | 5.03    | 0.17    | 4.86    | 0.08         | 2.77         | 97.36           | 0.25    | 144.45  | 3.86    | 335823.04 |         |
| <b>std</b>   | 515.98            | 0.83                | 5.23    | 0.65    | 5.13    | 0.45         | 1.86         | 49.84           | 1.03    | 458.11  | 2.75    | 417738.68 |         |
| <b>min</b>   | 0.00              | 0.00                | 0.00    | 0.00    | 0.00    | 0.00         | 1.00         | 5.00            | 0.00    | 0.00    | 2.00    | 380.00    |         |
| <b>25%</b>   | 12.00             | 0.00                | 1.00    | 0.00    | 1.00    | 0.00         | 1.00         | 38.00           | 0.00    | 4.00    | 2.00    | 12330.00  |         |
| <b>50%</b>   | 85.00             | 0.01                | 3.00    | 0.00    | 3.00    | 0.00         | 2.00         | 112.00          | 0.00    | 83.50   | 2.00    | 76599.00  |         |
| <b>75%</b>   | 554.00            | 0.03                | 9.00    | 0.00    | 9.00    | 0.00         | 4.00         | 140.00          | 0.00    | 214.00  | 8.00    | 976103.00 |         |
| <b>max</b>   | 3285.00           | 5.91                | 63.00   | 5.00    | 63.00   | 4.00         | 10.00        | 156.00          | 9.00    | 9999.00 | 9.00    | 999999.00 |         |

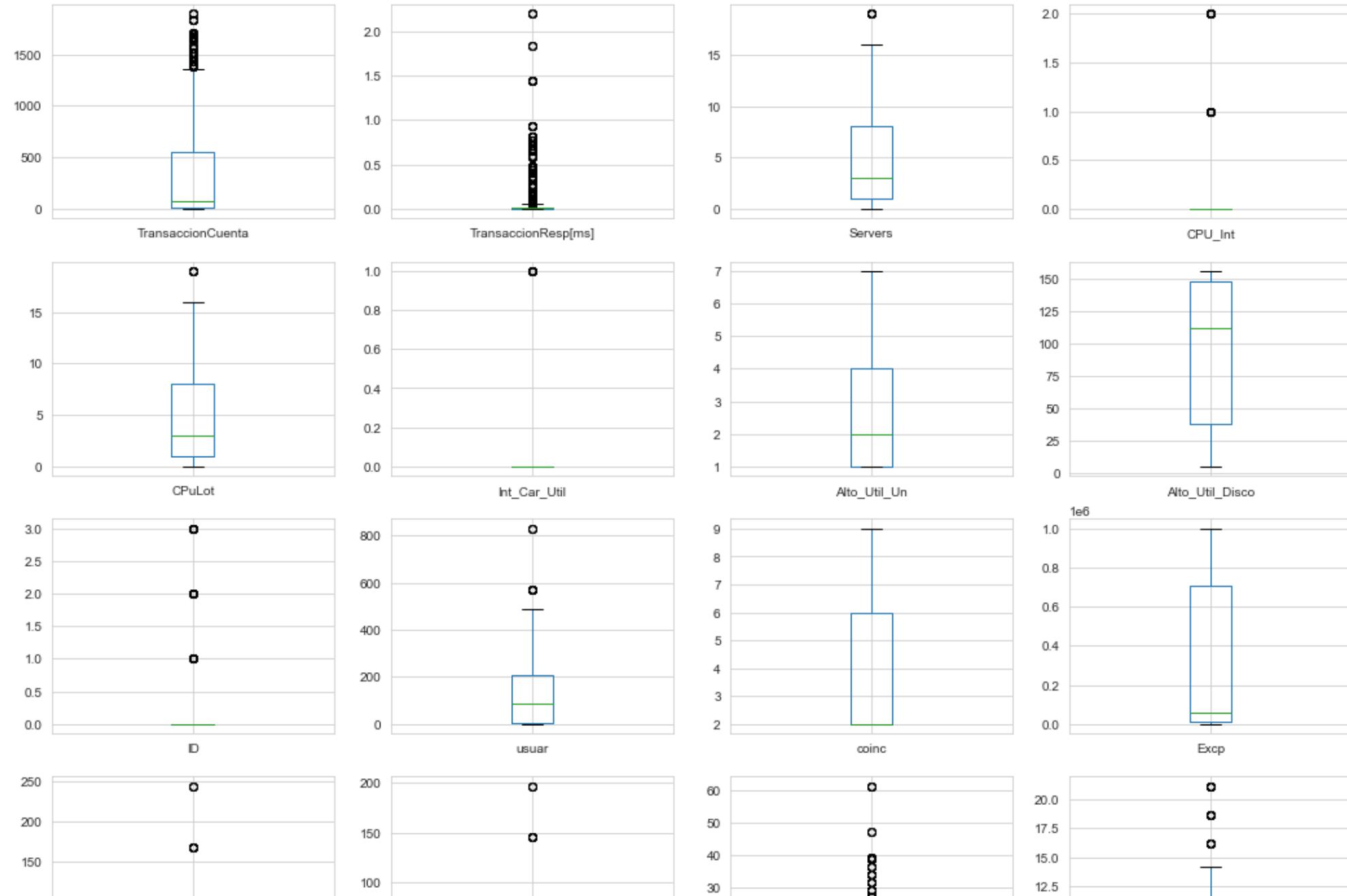
Registros Eliminados 460

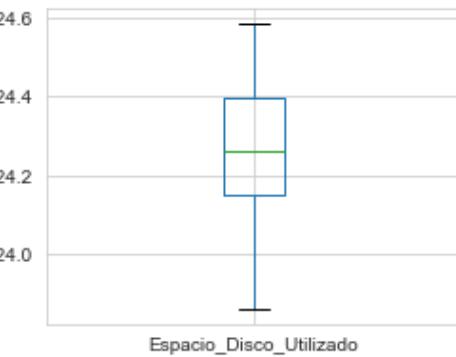
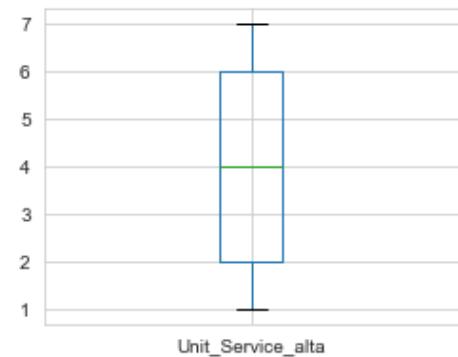
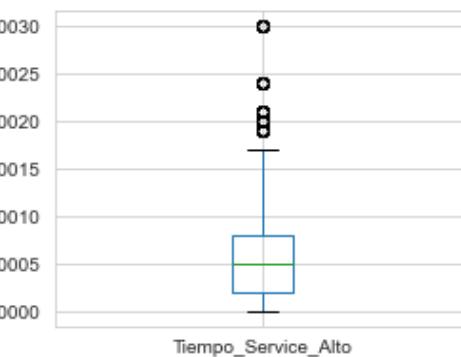
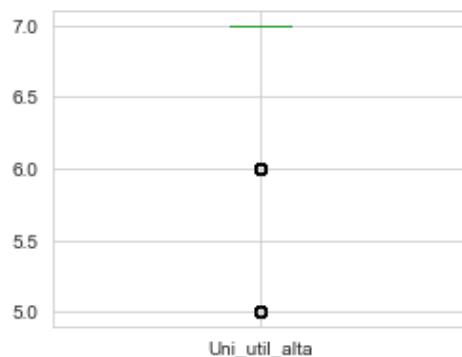
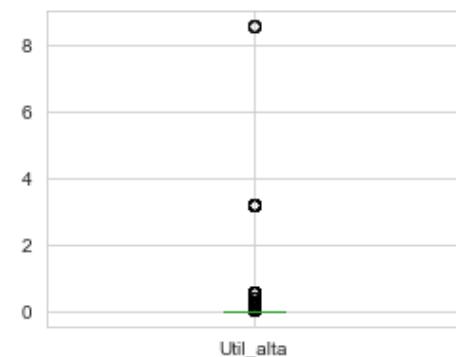
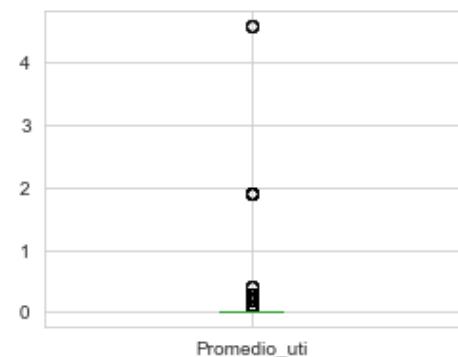
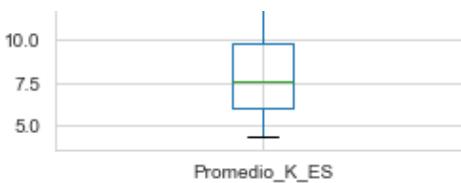
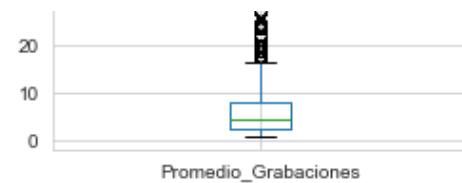
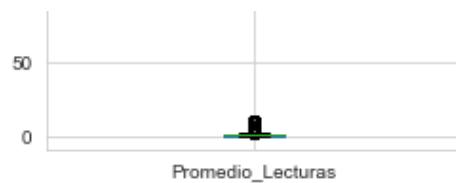
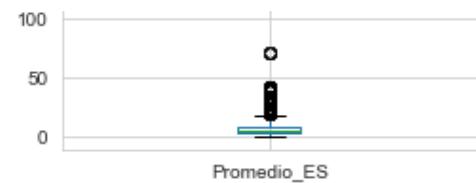
In [37]: df\_clai.head(5)

Out[37]:

|    |  | Fecha               | TransaccionCuenta | TransaccionResp[ms] | Servers | CPU_Int | CPuLot | Int_Car_Util | Alto_Util_Un | Alto_Util_Disco | ID | usuар | coinc | Excp   | Promedi |
|----|--|---------------------|-------------------|---------------------|---------|---------|--------|--------------|--------------|-----------------|----|-------|-------|--------|---------|
| 10 |  | 2021-03-12 00:55:00 | 36                | 0.02                | 1       | 0       | 1      | 0            | 4            | 156             | 0  | 133   | 2     | 20084  |         |
| 11 |  | 2021-03-12 01:00:00 | 0                 | 0.00                | 1       | 0       | 1      | 0            | 4            | 156             | 0  | 171   | 2     | 94267  |         |
| 13 |  | 2021-03-12 01:10:00 | 18                | 0.00                | 3       | 0       | 3      | 0            | 5            | 150             | 0  | 310   | 2     | 999999 |         |
| 14 |  | 2021-03-12 01:15:00 | 0                 | 0.00                | 4       | 0       | 4      | 0            | 6            | 156             | 0  | 250   | 2     | 999999 |         |
| 15 |  | 2021-03-12 01:20:00 | 0                 | 0.00                | 4       | 0       | 4      | 0            | 4            | 148             | 0  | 175   | 2     | 999999 |         |

```
In [38]: # Visualizamos rápidamente las características de entrada  
df_clai.select_dtypes(exclude = ['datetime', 'object']).plot(kind='box', subplots=True, layout=(7,4), figsize=(18,25), sharex=False)  
plt.show()
```





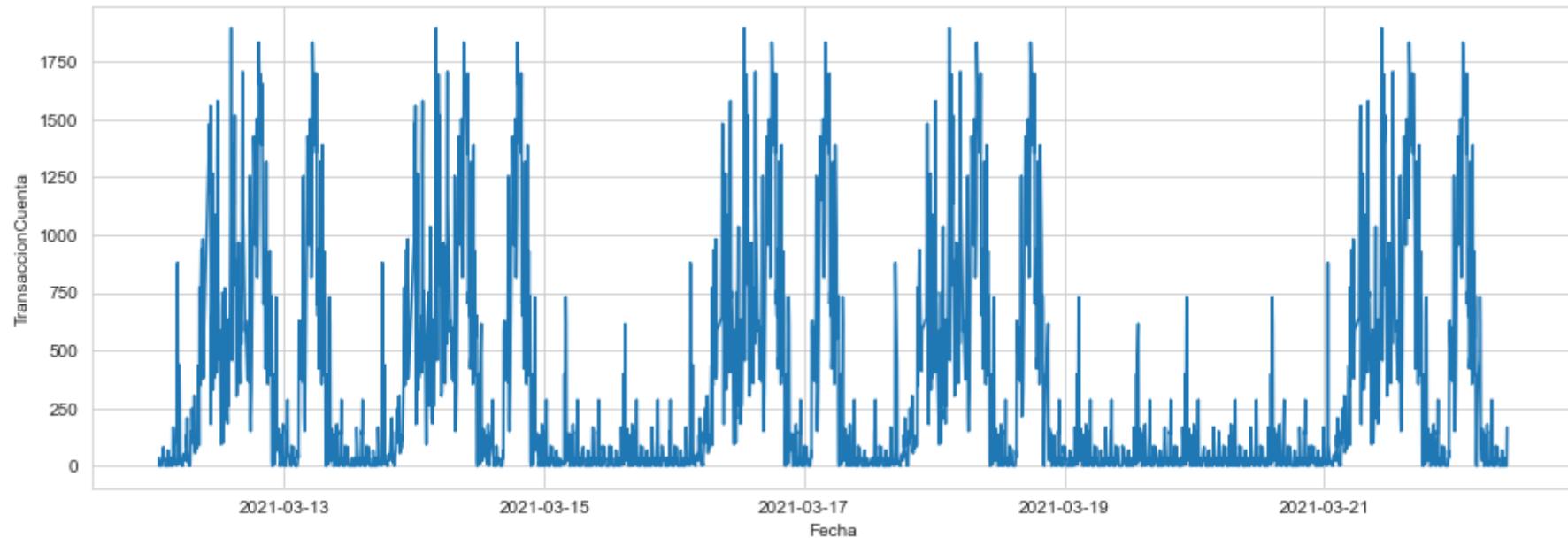
In [39]:

```
# -----  
# DataFrame to excel  
# ref: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to\_excel.html  
# -----  
  
df_sheet_1 = df_trx.copy()  
df_sheet_1_name = "Data_Transaccional"  
  
df_sheet_2 = df_hrd.copy()  
df_sheet_2_name = "Data_Hardware"  
  
df_sheet_3 = df_clai.copy()  
df_sheet_3_name = "Data_Unificada"  
  
with pd.ExcelWriter('Datasets\output.xlsx') as writer:  
    df_sheet_1.to_excel(writer, index=False, sheet_name=df_sheet_1_name)  
    df_sheet_2.to_excel(writer, index=False, sheet_name=df_sheet_2_name)  
    df_sheet_3.to_excel(writer, index=False, sheet_name=df_sheet_3_name)
```

In [40]:

```
sns.lineplot(data=df_clai, x="Fecha", y="TransaccionCuenta")
```

Out[40]: <AxesSubplot:xlabel='Fecha', ylabel='TransaccionCuenta'>



# 5 MONGO DB

## 5.1 pymongo

```
In [41]: # # https://jakevdp.github.io/blog/2017/12/05/installing-python-packages-from-jupyter/
# # Install a conda package in the current Jupyter kernel
# # conda install -c conda-forge win10toast
# !conda install --yes --prefix f'{sys.prefix}' pymongo[srv]
# !conda install --yes --prefix {sys.prefix} dnspython

# conda install -c conda-forge --yes pymongo[srv]
```

## 5.2 Mongodb Atlas

MongoDB Atlas proporciona una manera fácil de alojar y administrar sus datos en la nube.

Guía para la creación de un clúster Atlas, la conexión a él, la inserción de datos y la consulta de datos. Get Started with Atlas:

<https://docs.atlas.mongodb.com/getting-started/> (<https://docs.atlas.mongodb.com/getting-started/>)

- <https://cloud.mongodb.com/v2/60d5b54b676f65672fc27c05#metrics/replicaSet/60d5bb01b504795e3c4ce263/explorer/lfcontrol/metrics/find>  
(<https://cloud.mongodb.com/v2/60d5b54b676f65672fc27c05#metrics/replicaSet/60d5bb01b504795e3c4ce263/explorer/lfcontrol/metrics/find>)

## 5.3 Conexión a la base de datos no relacional Mongo DB

```
In [42]: try:
    from decouple import config
    from pymongo import MongoClient

    # Usuario y contraseña para la conexión a la base de datos MongoDB
    db_user = config('MONGO_USER', default='user')
    db_pass = config('MONGO_PASS', default='pass')

    # Nombre de la base de datos MongoDB
    db_name = 'lfControlDatabase'

    # URL de conexión
    mongodb_connection = f"mongodb+srv://{{db_user}}:{{db_pass}}@lfcontrolcluster.yiru3.mongodb.net/{{db_name}}?retryWrites=true&w=majority"

    # Conexión
    client = MongoClient(mongodb_connection)

    # Database LfControl
    db = client.lfcontrol

    # Obtener métricas de una colección
    sel_collection = "metrics"
    metrics_collection = db[sel_collection]
    #     metrics_collection = db.metrics

    #     # Getting dbs names
    #     display(f"DBs: {list(client.list_databases())}" )

    # Getting Collections names
    display(f"Colecciones: {str(db.list_collection_names())}" )

    # Count_documents
    display(f"Número documentos en colección \"{{sel_collection}}\": {str(metrics_collection.count_documents({}))}" )

except ImportError:
    print("pip install python-decouple")
    print("pip install pymongo[srv]")
```

"Colecciones: ['test', 'metrics']"

'Número documentos en colección "metrics": 2540'

## **5.4 Conversión de la data a formato JSon para el almacenamiento en MongoDB**

In [43]:

```
import json

# Convertir df a Json
temp_df = df_clai.copy()
result = temp_df.to_json(orient="records", date_format='iso', date_unit='s')

# Convertir cadena text JSON en un diccionario Python
parsed = json.loads(result)

# Convertir diccionario Python en un elemnto JSON :
json_data = json.dumps(parsed[:1], indent=2)

print(json_data)
```

```
[  
  {  
    "Fecha": "2021-03-12T00:55:00Z",  
    "TransaccionCuenta": 36,  
    "TransaccionResp[ms)": 0.02,  
    "Servers": 1,  
    "CPU_Int": 0,  
    "CPuLot": 1,  
    "Int_Car_Util": 0,  
    "Alto_Util_Un": 4,  
    "Alto_Util_Disco": 156,  
    "ID": 0,  
    "usuar": 133,  
    "coinc": 2,  
    "Excp": 20084,  
    "Promedio_ES": 244.1,  
    "Promedio_Lecturas": 196.8,  
    "Promedio_Grabaciones": 47.3,  
    "Promedio_K_ES": 9.5,  
    "Promedio_utí": 4.6,  
    "Util_alta": 8.6,  
    "Uni_util_alta": 7,  
    "Tiempo_Service_Alto": 0.0019,  
    "Unit_Service_alta": 7,  
    "Espacio_Disco_Utilizado": 823.885  
  }  
]
```

## 5.5 Almacenamiento de los datos en Base de datos Mongo DB

```
In [44]: # https://pymongo.readthedocs.io/en/stable/api/pymongo/collection.html#pymongo.collection.Collection.insert_one

# Check Environment Test
try:
    from decouple import config
    if not config('ENV_PRD', default=False, cast=bool):
        # Remove all docs in the db
        metrics_collection.remove({})
except:
    metrics_collection.remove({})

# Insert documents in db
def write_db(collection=db.test):
    #     db_result = metrics_collection.insert_many(parsed)
    db_result = ''
    try:
        db_result = collection.insert_many(parsed)
        print(f"Se han insertado {collection.count_documents({})} documentos")
    except:
        print("Unexpected error:", sys.exc_info()[0], str(db_result))
        raise
    return db_result

# save data in json format (mongodb database)
write_db(collection=metrics_collection)
```

Se han insertado 2540 documentos

Out[44]: <pymongo.results.InsertManyResult at 0x216d628f480>

```
In [45]: print(metrics_collection.count_documents({}))
```

2540

## 5.6 Importar datos Formato JSON

In [46]:

```
import pprint
from bson.json_util import dumps

cursor = metrics_collection.find({})
data = list(cursor)

print_data = dumps(data[:1], indent=2)
print(print_data)
```

```
[  
  {  
    "_id": {  
      "$oid": "613f7c9724369324338c66f9"  
    },  
    "Fecha": "2021-03-12T00:55:00Z",  
    "TransaccionCuenta": 36,  
    "TransaccionResp[ms)": 0.02,  
    "Servers": 1,  
    "CPU_Int": 0,  
    "CPULot": 1,  
    "Int_Car_Util": 0,  
    "Alto_Util_Un": 4,  
    "Alto_Util_Disco": 156,  
    "ID": 0,  
    "usuari": 133,  
    "coinc": 2,  
    "Excp": 20084,  
    "Promedio_ES": 244.1,  
    "Promedio_Lecturas": 196.8,  
    "Promedio_Grabaciones": 47.3,  
    "Promedio_K_ES": 9.5,  
    "Promedio_utí": 4.6,  
    "Util_alta": 8.6,  
    "Uni_utí_alta": 7,  
    "Tiempo_Service_Alto": 0.0019,  
    "Unit_Service_alta": 7,  
    "Espacio_Disco_Utilizado": 823.885  
  }  
]
```

## 6 Generación de visualizaciones

```
In [47]: print(list(df_clai.columns))
```

```
['Fecha', 'TransaccionCuenta', 'TransaccionResp[ms]', 'Servers', 'CPU_Int', 'CPuLot', 'Int_Car_Util', 'Alto_Util_Un', 'Alto_Util_Disco', 'ID', 'usuari', 'coinc', 'Excp', 'Promedio_ES', 'Promedio_Lecturas', 'Promedio_Grabaciones', 'Promedio_K_ES', 'Promedio_ut_i', 'Util_alta', 'Uni_util_alta', 'Tiempo_Service_Alto', 'Unit_Service_alta', 'Espacio_Disco_Utilizado']
```

In [48]:

```
import math

sns.set_theme(style="darkgrid")
# feature_list = ['TransaccionCuenta', 'TransaccionResp[ms]', 'Servers', 'CPU_Int',
#                 'CPuLot', 'Int_Car_Util', 'Alto_Util_Un', 'Alto_Util_Disco', 'ID',
#                 'usuari', 'coinc', 'Excp', 'Promedio_ES',
#                 'Promedio_Lecturas', 'Promedio_Grabaciones', 'Promedio_K_ES',
#                 'Promedio_uti', 'Util_alta', 'Uni_util_alta', 'Tiempo_Service_Alto',
#                 'Unit_Service_alta', 'Espacio_Disco_Utilizado'
#                 ]
#             ]

feature_list = list(df_clai.select_dtypes(exclude = ['datetime', 'object'] ).columns)

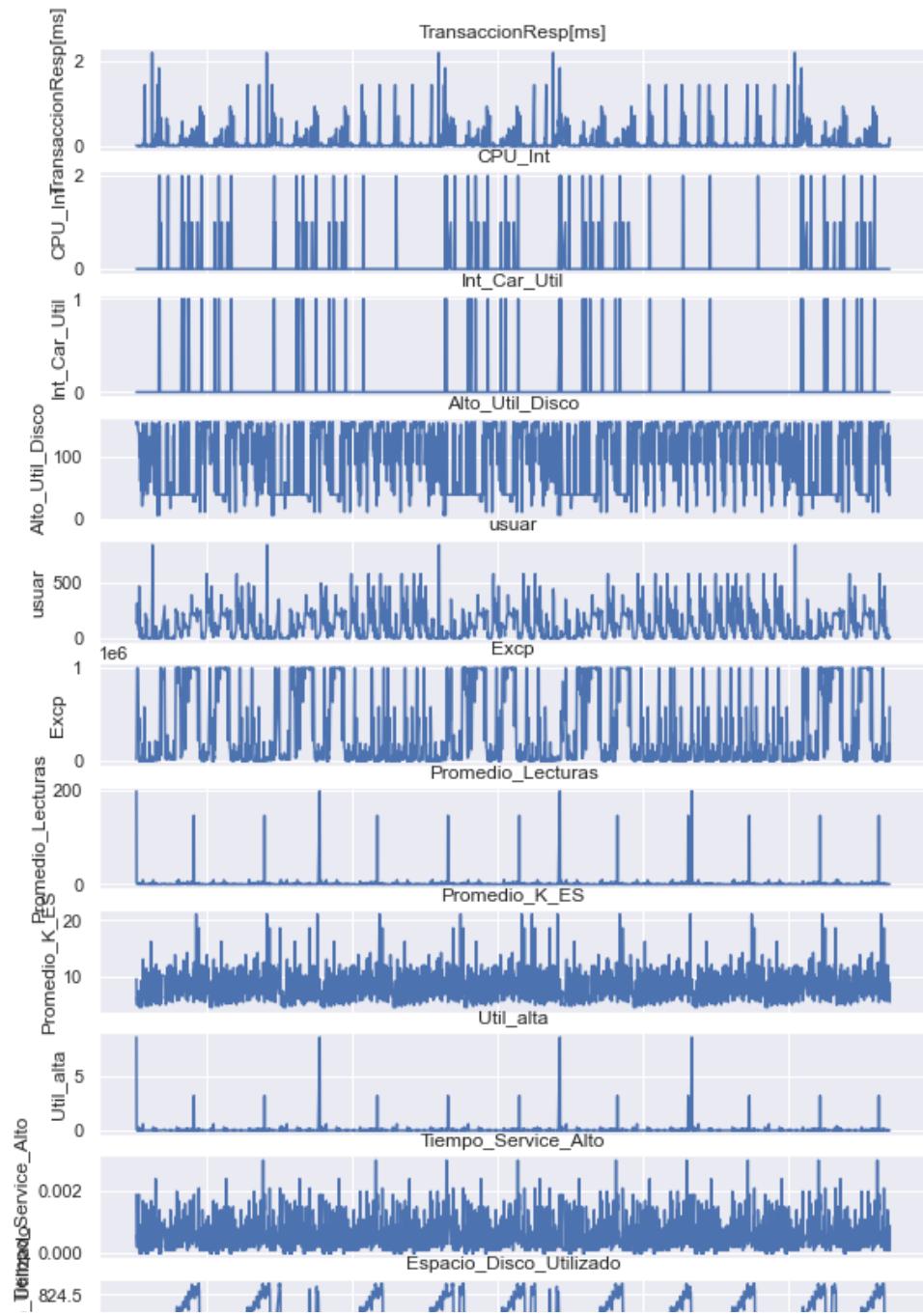
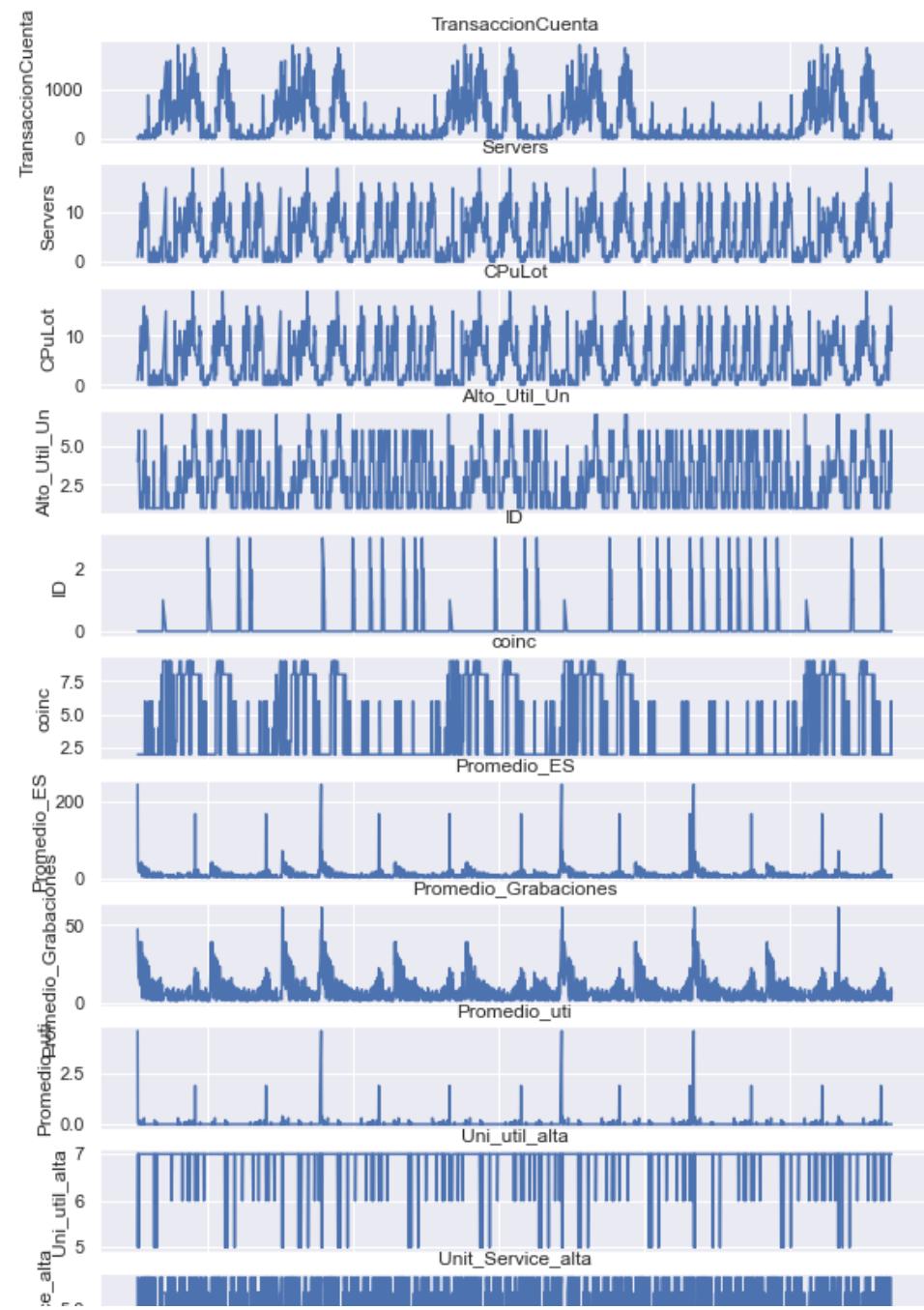
n_rows, n_cols = (math.ceil(len(feature_list)/2), 2) # Fila y Columnas

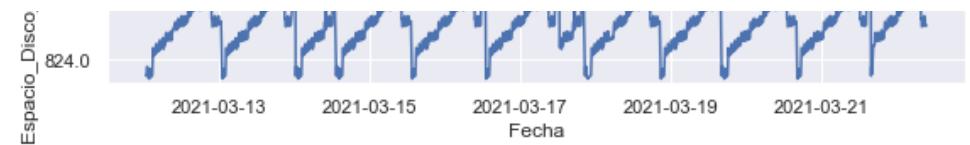
if n_rows >= 2:
    fig, axes = plt.subplots(n_rows, n_cols, sharex=True, figsize=(20,15))
else:
    fig, axes = plt.subplots(n_cols,n_rows, sharex=True, figsize=(20,15))
fig.suptitle('Data')

variable_x = "Fecha"
# variable_x = "tiempo"

if n_rows >= 2:
    for i in range(n_rows):
        for j in range(n_cols):
            if len(feature_list) > 0:
                sns.lineplot(ax=axes[i, j], data=df_clai, x=variable_x, y=feature_list[0])
                axes[i, j].set_title(feature_list[0])
                feature_list.pop(0)
            else:
                break
    else:
        for i in range(n_cols):
            if len(feature_list) > 0:
                sns.lineplot(ax=axes[i], data=df_clai, x=variable_x, y=feature_list[0])
                axes[i].set_title(feature_list[0])
                feature_list.pop(0)
            else:
                break
```

Data



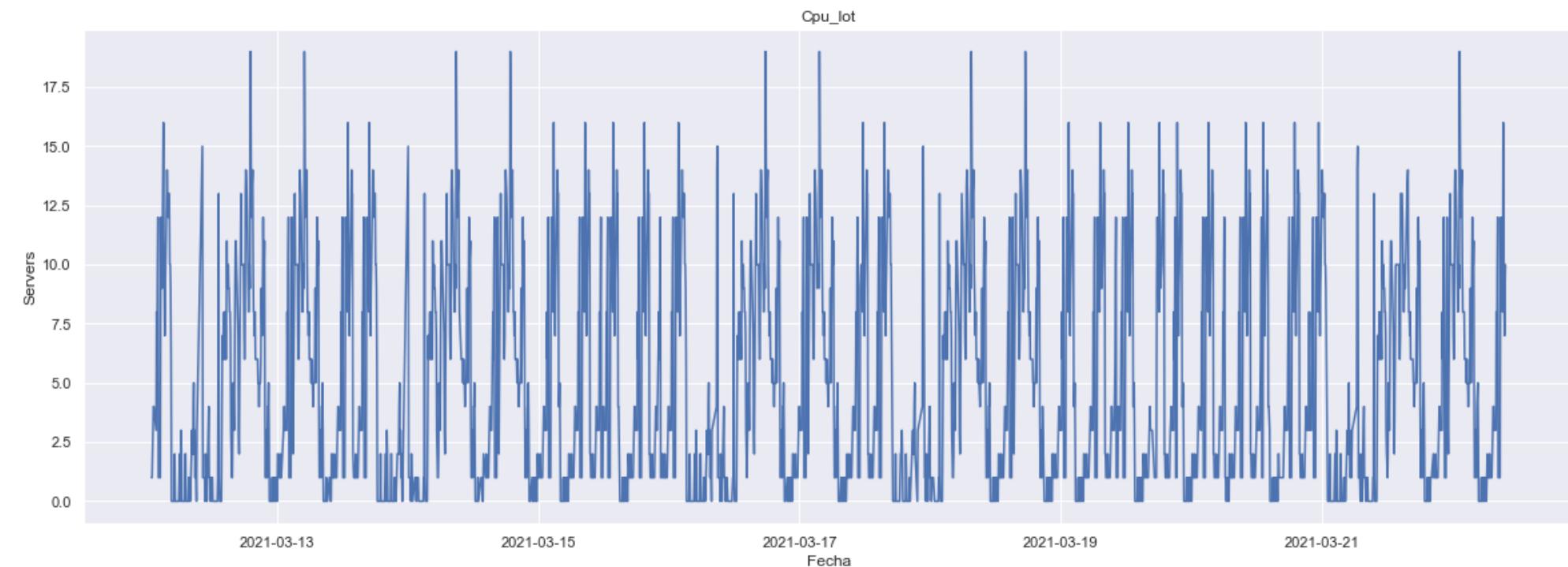
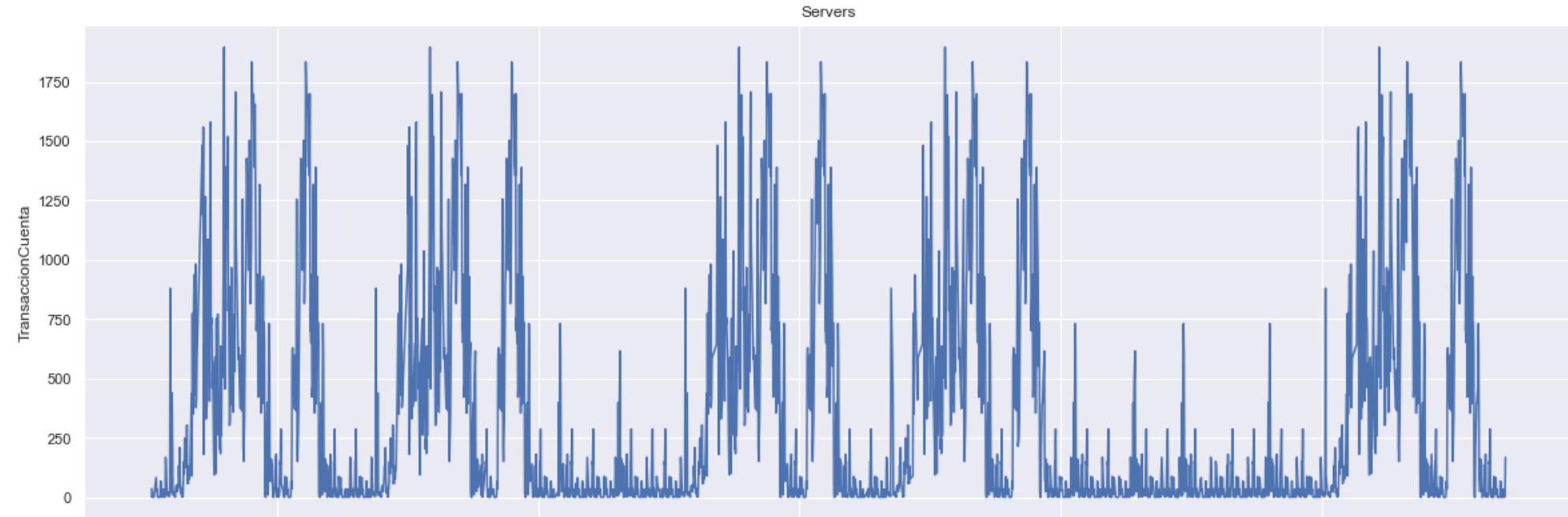


```
In [49]: fig, axes = plt.subplots(2,1, sharex=True, figsize=(20,15))

sns.lineplot(data=df_clai, x=variable_x, y="TransaccionCuenta", ax=axes[0])
axes[0].set_title("Servers")

sns.lineplot(data=df_clai, x=variable_x, y="Servers", ax=axes[1] )
axes[1].set_title("Cpu_lot")
```

```
Out[49]: Text(0.5, 1.0, 'Cpu_lot')
```



```
In [50]: print(list(df_clai.columns))
```

```
['Fecha', 'TransaccionCuenta', 'TransaccionResp[ms]', 'Servers', 'CPU_Int', 'CPuLot', 'Int_Car_Util', 'Alto_Util_Un', 'Alto_Util_Disco', 'ID', 'usuar', 'coinc', 'Excp', 'Promedio_ES', 'Promedio_Lecturas', 'Promedio_Grabaciones', 'Promedio_K_ES', 'Promedio_ut_i', 'Util_alta', 'Uni_util_alta', 'Tiempo_Service_Alto', 'Unit_Service_alta', 'Espacio_Disco_Utilizado']
```

In [51]:

```
# Crea index de agrupacion
df_clai['g_index'] = [1, 2, 3, 4, 5] * int(df_clai.shape[0]/5)

fun = ['sum', 'mean', 'median', 'min', 'max',]

gdf = df_clai.groupby(['g_index']).agg(
    {'TransaccionCuenta': fun,
     'TransaccionResp[ms]': fun,
     'Servers': fun,
     'CPU_Int': fun,
     'CPuLot': fun,
     'Int_Car_Util': fun,
     'Alto_Util_Un': fun,
     'Alto_Util_Disco': fun,
     'ID': fun,
     'usuar': fun,
     'coinc': fun,
     'Excp': fun,
     'Promedio_ES': fun,
     'Promedio_Lecturas': fun,
     'Promedio_Grabaciones': fun,
     'Promedio_K_ES': fun,
     'Promedio_utí': fun,
     'Util_alta': fun,
     'Uni_util_alta': fun,
     'Tiempo_Service_Alto': fun,
     'Unit_Service_alta': fun,
     'Espacio_Disco_Utilizado': fun
})#.reset_index()
gdf
```

Out[51]:

| g_index | TransaccionCuenta |         |        |     |      | TransaccionResp[ms] |       |        |     |      | Servers |       |        |     |     | ... |       |           |           |     | Tiempo_Service_Alto |     |      |        |     | U   |    |
|---------|-------------------|---------|--------|-----|------|---------------------|-------|--------|-----|------|---------|-------|--------|-----|-----|-----|-------|-----------|-----------|-----|---------------------|-----|------|--------|-----|-----|----|
|         | sum               | mean    | median | min | max  | sum                 | mean  | median | min | max  | sum     | mean  | median | min | max | ... | sum   | mean      | median    | min | max                 | sum | mean | median | min | max | su |
| 1       | 161364            | 317.646 | 74.5   | 0   | 1895 | 41.92               | 0.083 | 0.01   | 0.0 | 2.20 | 2260    | 4.449 | 3      | 0   | 19  | ... | 0.319 | 6.285e-04 | 5.000e-04 | 0.0 | 0.003               | 20  |      |        |     |     |    |
| 2       | 174505            | 343.514 | 78.0   | 0   | 1895 | 32.93               | 0.065 | 0.01   | 0.0 | 1.44 | 2374    | 4.673 | 3      | 0   | 19  | ... | 0.296 | 5.827e-04 | 4.000e-04 | 0.0 | 0.003               | 20  |      |        |     |     |    |
| 3       | 172754            | 340.067 | 76.0   | 0   | 1833 | 33.11               | 0.065 | 0.01   | 0.0 | 2.20 | 2366    | 4.657 | 3      | 0   | 16  | ... | 0.308 | 6.067e-04 | 5.000e-04 | 0.0 | 0.003               | 20  |      |        |     |     |    |

| g_index | TransaccionCuenta |         |        |     |      | TransaccionResp[ms] |       |        |     |      | Servers |       |        |     |     | ... |       |           |           |     | Tiempo_Service_Alto |     |     |      |        | U   |     |     |
|---------|-------------------|---------|--------|-----|------|---------------------|-------|--------|-----|------|---------|-------|--------|-----|-----|-----|-------|-----------|-----------|-----|---------------------|-----|-----|------|--------|-----|-----|-----|
|         | sum               | mean    | median | min | max  | sum                 | mean  | median | min | max  | sum     | mean  | median | min | max | ... | sum   | mean      | median    | min | max                 | ... | sum | mean | median | min | max | sum |
| 4       | 165482            | 325.752 | 83.0   | 0   | 1833 | 33.99               | 0.067 | 0.01   | 0.0 | 2.20 | 2392    | 4.709 | 3      | 0   | 19  | ... | 0.305 | 6.008e-04 | 5.000e-04 | 0.0 | 0.003               | 20  |     |      |        |     |     |     |
| 5       | 161151            | 317.226 | 78.0   | 0   | 1895 | 35.53               | 0.070 | 0.01   | 0.0 | 1.84 | 2308    | 4.543 | 3      | 0   | 19  | ... | 0.299 | 5.884e-04 | 4.000e-04 | 0.0 | 0.003               | 20  |     |      |        |     |     |     |

5 rows × 110 columns

In [52]:

```
kys= list(set([list(gdf.keys())[x]][0] for x in range(len(list(list(gdf.keys()))))]))  
# display(kys)  
  
df_collection = {}  
for el in kys:  
    df_collection[el] = pd.DataFrame(gdf[el].reset_index()).drop(['g_index'], axis = 1)  
    print(el, "\n"*2, df_collection[el], "\n"*2)  
  
# col_el = 'Alto_Util_Un'  
# display(col_el, df_collection[col_el])
```

Espacio\_Disco\_Utilizado

|   | sum        | mean    | median  | min     | max     |
|---|------------|---------|---------|---------|---------|
| 0 | 418725.210 | 824.262 | 824.260 | 823.857 | 824.588 |
| 1 | 418724.771 | 824.261 | 824.259 | 823.857 | 824.588 |
| 2 | 418725.208 | 824.262 | 824.260 | 823.858 | 824.588 |
| 3 | 418724.706 | 824.261 | 824.259 | 823.857 | 824.588 |
| 4 | 418724.879 | 824.262 | 824.259 | 823.857 | 824.588 |

usuari

|   | sum   | mean    | median | min | max |
|---|-------|---------|--------|-----|-----|
| 0 | 57443 | 113.077 | 79.0   | 0   | 829 |
| 1 | 59505 | 117.136 | 79.0   | 0   | 572 |
| 2 | 59849 | 117.813 | 85.0   | 0   | 829 |
| 3 | 63429 | 124.860 | 86.5   | 0   | 572 |
| 4 | 60583 | 119.258 | 84.0   | 0   | 829 |

Int\_Car\_Util

|   | sum | mean  | median | min | max |
|---|-----|-------|--------|-----|-----|
| 0 | 7   | 0.014 | 0      | 0   | 1   |
| 1 | 10  | 0.020 | 0      | 0   | 1   |
| 2 | 10  | 0.020 | 0      | 0   | 1   |
| 3 | 15  | 0.030 | 0      | 0   | 1   |
| 4 | 14  | 0.028 | 0      | 0   | 1   |

Tiempo\_Service\_Alto

|   | sum   | mean      | median    | min | max   |
|---|-------|-----------|-----------|-----|-------|
| 0 | 0.319 | 6.285e-04 | 5.000e-04 | 0.0 | 0.003 |
| 1 | 0.296 | 5.827e-04 | 4.000e-04 | 0.0 | 0.003 |
| 2 | 0.308 | 6.067e-04 | 5.000e-04 | 0.0 | 0.003 |
| 3 | 0.305 | 6.008e-04 | 5.000e-04 | 0.0 | 0.003 |
| 4 | 0.299 | 5.884e-04 | 4.000e-04 | 0.0 | 0.003 |

#### Unit\_Service\_alta

|   | sum  | mean  | median | min | max |
|---|------|-------|--------|-----|-----|
| 0 | 2026 | 3.988 | 4      | 1   | 7   |
| 1 | 2088 | 4.110 | 4      | 1   | 7   |
| 2 | 2048 | 4.031 | 4      | 1   | 7   |
| 3 | 2011 | 3.959 | 4      | 1   | 7   |
| 4 | 2015 | 3.967 | 4      | 1   | 7   |

#### ID

|   | sum | mean  | median | min | max |
|---|-----|-------|--------|-----|-----|
| 0 | 78  | 0.154 | 0      | 0   | 3   |
| 1 | 79  | 0.156 | 0      | 0   | 3   |
| 2 | 87  | 0.171 | 0      | 0   | 3   |
| 3 | 86  | 0.169 | 0      | 0   | 3   |
| 4 | 93  | 0.183 | 0      | 0   | 3   |

#### TransaccionResp[ms]

|   | sum   | mean  | median | min | max  |
|---|-------|-------|--------|-----|------|
| 0 | 41.92 | 0.083 | 0.01   | 0.0 | 2.20 |
| 1 | 32.93 | 0.065 | 0.01   | 0.0 | 1.44 |
| 2 | 33.11 | 0.065 | 0.01   | 0.0 | 2.20 |
| 3 | 33.99 | 0.067 | 0.01   | 0.0 | 2.20 |
| 4 | 35.53 | 0.070 | 0.01   | 0.0 | 1.84 |

#### CPU\_Int

|   | sum | mean  | median | min | max |
|---|-----|-------|--------|-----|-----|
| 0 | 35  | 0.069 | 0      | 0   | 2   |
| 1 | 34  | 0.067 | 0      | 0   | 2   |
| 2 | 34  | 0.067 | 0      | 0   | 2   |

|   |    |       |   |   |   |
|---|----|-------|---|---|---|
| 3 | 41 | 0.081 | 0 | 0 | 2 |
| 4 | 46 | 0.091 | 0 | 0 | 2 |

### CPuLot

|   |      | sum   | mean | median | min | max |
|---|------|-------|------|--------|-----|-----|
| 0 | 2225 | 4.380 |      | 3      | 0   | 19  |
| 1 | 2340 | 4.606 |      | 3      | 0   | 19  |
| 2 | 2332 | 4.591 |      | 3      | 0   | 16  |
| 3 | 2351 | 4.628 |      | 3      | 0   | 19  |
| 4 | 2262 | 4.453 |      | 3      | 0   | 19  |

### Promedio\_Lecturas

|   |        | sum   | mean | median | min | max   |
|---|--------|-------|------|--------|-----|-------|
| 0 | 896.0  | 1.764 |      | 0.10   | 0.0 | 196.8 |
| 1 | 518.7  | 1.021 |      | 0.15   | 0.0 | 145.1 |
| 2 | 1187.6 | 2.338 |      | 0.10   | 0.0 | 196.8 |
| 3 | 633.4  | 1.247 |      | 0.10   | 0.0 | 145.1 |
| 4 | 819.6  | 1.613 |      | 0.20   | 0.0 | 145.1 |

### Util\_alta

|   |      | sum   | mean | median | min | max |
|---|------|-------|------|--------|-----|-----|
| 0 | 31.5 | 0.062 |      | 0.0    | 0.0 | 8.6 |
| 1 | 14.4 | 0.028 |      | 0.0    | 0.0 | 3.2 |
| 2 | 36.0 | 0.071 |      | 0.0    | 0.0 | 8.6 |
| 3 | 17.5 | 0.034 |      | 0.0    | 0.0 | 3.2 |
| 4 | 20.8 | 0.041 |      | 0.0    | 0.0 | 3.2 |

### coinc

|   |      | sum   | mean | median | min | max |
|---|------|-------|------|--------|-----|-----|
| 0 | 1911 | 3.762 |      | 2      | 2   | 9   |
| 1 | 1881 | 3.703 |      | 2      | 2   | 9   |
| 2 | 1922 | 3.783 |      | 2      | 2   | 9   |
| 3 | 1935 | 3.809 |      | 2      | 2   | 9   |
| 4 | 1942 | 3.823 |      | 2      | 2   | 9   |

Excp

|   | sum       | mean       | median  | min | max    |
|---|-----------|------------|---------|-----|--------|
| 0 | 154822095 | 304767.904 | 60193.0 | 380 | 999999 |
| 1 | 156466474 | 308004.870 | 54828.0 | 380 | 999999 |
| 2 | 153846249 | 302846.947 | 51227.0 | 380 | 999999 |
| 3 | 158955687 | 312904.896 | 55264.0 | 380 | 999999 |
| 4 | 155709536 | 306514.835 | 72592.5 | 380 | 999999 |

Servers

|   | sum  | mean  | median | min | max |
|---|------|-------|--------|-----|-----|
| 0 | 2260 | 4.449 | 3      | 0   | 19  |
| 1 | 2374 | 4.673 | 3      | 0   | 19  |
| 2 | 2366 | 4.657 | 3      | 0   | 16  |
| 3 | 2392 | 4.709 | 3      | 0   | 19  |
| 4 | 2308 | 4.543 | 3      | 0   | 19  |

Alto\_Util\_Un

|   | sum  | mean  | median | min | max |
|---|------|-------|--------|-----|-----|
| 0 | 1296 | 2.551 | 2      | 1   | 7   |
| 1 | 1329 | 2.616 | 2      | 1   | 7   |
| 2 | 1317 | 2.593 | 2      | 1   | 7   |
| 3 | 1317 | 2.593 | 2      | 1   | 7   |
| 4 | 1326 | 2.610 | 2      | 1   | 7   |

Promedio\_utl

|   | sum  | mean  | median | min | max |
|---|------|-------|--------|-----|-----|
| 0 | 17.1 | 0.034 | 0.0    | 0.0 | 4.6 |
| 1 | 6.4  | 0.013 | 0.0    | 0.0 | 1.9 |
| 2 | 19.7 | 0.039 | 0.0    | 0.0 | 4.6 |
| 3 | 9.2  | 0.018 | 0.0    | 0.0 | 1.9 |
| 4 | 11.1 | 0.022 | 0.0    | 0.0 | 1.9 |

Promedio\_Grabaciones

|   | sum    | mean  | median | min | max  |
|---|--------|-------|--------|-----|------|
| 0 | 3148.8 | 6.198 | 3.9    | 0.9 | 61.2 |

|   |        |       |     |     |      |
|---|--------|-------|-----|-----|------|
| 1 | 3529.4 | 6.948 | 4.7 | 0.9 | 39.0 |
| 2 | 3182.0 | 6.264 | 3.9 | 0.9 | 61.2 |
| 3 | 3181.0 | 6.262 | 4.4 | 0.9 | 39.0 |
| 4 | 3499.6 | 6.889 | 4.7 | 0.9 | 61.2 |

#### Promedio\_K\_ES

|   | sum    | mean  | median | min | max  |
|---|--------|-------|--------|-----|------|
| 0 | 4214.2 | 8.296 | 8.1    | 4.4 | 21.1 |
| 1 | 4058.8 | 7.990 | 7.3    | 4.4 | 21.1 |
| 2 | 4166.8 | 8.202 | 7.8    | 4.4 | 21.1 |
| 3 | 4064.8 | 8.002 | 7.3    | 4.4 | 21.1 |
| 4 | 4104.7 | 8.080 | 7.8    | 4.4 | 21.1 |

#### Alto\_Util\_Disco

|   | sum   | mean    | median | min | max |
|---|-------|---------|--------|-----|-----|
| 0 | 50032 | 98.488  | 112.0  | 5   | 156 |
| 1 | 51467 | 101.313 | 126.0  | 5   | 156 |
| 2 | 50109 | 98.640  | 123.0  | 5   | 156 |
| 3 | 49035 | 96.526  | 110.5  | 5   | 156 |
| 4 | 50514 | 99.437  | 112.0  | 5   | 156 |

#### Promedio\_ES

|   | sum    | mean  | median | min | max   |
|---|--------|-------|--------|-----|-------|
| 0 | 4067.2 | 8.006 | 4.50   | 1.0 | 244.1 |
| 1 | 4072.4 | 8.017 | 5.50   | 1.0 | 167.3 |
| 2 | 4392.5 | 8.647 | 4.35   | 1.0 | 244.1 |
| 3 | 3838.1 | 7.555 | 4.90   | 1.0 | 167.3 |
| 4 | 4342.4 | 8.548 | 5.40   | 1.0 | 167.3 |

#### Uni\_util\_alta

|   | sum  | mean  | median | min | max |
|---|------|-------|--------|-----|-----|
| 0 | 3533 | 6.955 | 7      | 5   | 7   |
| 1 | 3514 | 6.917 | 7      | 5   | 7   |
| 2 | 3533 | 6.955 | 7      | 5   | 7   |
| 3 | 3519 | 6.927 | 7      | 5   | 7   |
| 4 | 3519 | 6.927 | 7      | 5   | 7   |

```
TransaccionCuenta
```

|   | sum    | mean    | median | min | max  |
|---|--------|---------|--------|-----|------|
| 0 | 161364 | 317.646 | 74.5   | 0   | 1895 |
| 1 | 174505 | 343.514 | 78.0   | 0   | 1895 |
| 2 | 172754 | 340.067 | 76.0   | 0   | 1833 |
| 3 | 165482 | 325.752 | 83.0   | 0   | 1833 |
| 4 | 161151 | 317.226 | 78.0   | 0   | 1895 |

```
In [53]: #Se eliminan algunas columnas  
no_utiles = ["g_index"]  
df_clai = delete_columns(_df=df_clai, del_cols=no_utiles)
```

Se eliminaron satisfactoriamente las columnas del dataset.

Las siguientes campos no existen como columnas en el dataset:

```
-----  
['g_index']
```

```
In [54]: print(list(df_clai.columns))
```

```
['Fecha', 'TransaccionCuenta', 'TransaccionResp[ms]', 'Servers', 'CPU_Int', 'CPuLot', 'Int_Car_Util', 'Alto_Util_Un', 'Alto_Util_Disco', 'ID', 'usuari', 'coinc', 'Excp', 'Promedio_ES', 'Promedio_Lecturas', 'Promedio_Grabaciones', 'Promedio_K_ES', 'Promedio_ut_i', 'Util_alta', 'Uni_util_alta', 'Tiempo_Service_Alto', 'Unit_Service_alta', 'Espacio_Disco_Utilizado']
```

```
In [55]: # -----  
# ref: https://echarts.apache.org/examples/en/editor.html?c=line-simple  
# ref: https://nicoohlri.gitlab.io/ipychart/user_guide/charts.html#line  
# -----  
try:  
    from ipychart import Chart  
except ModuleNotFoundError as e:  
    print(e)  
    print("\nInstalar modulo: ->\tpip install ipychart")
```

```
In [56]: # !pip install ipychart
```

In [57]:

```
# https://nicohlr.gitlab.io/ipychart/user_guide/scales.html#time-argument

def ipychart_line(chr_df, x, y, title=None):
    dataset = {
        'labels': list(chr_df[x]),
        'datasets': [{ 'data': list(chr_df[y]), 'type': 'line',
                      'pointRadius': 0.1, 'pointHoverRadius': 10, 'borderWidth': 1
                    }]
    }
    options = {
        'title': {
            'display': True,
            'text': title,
            'fontSize': 30,
            'fontStyle': 'bold',
            'layout': {
                'padding': {'left': 40, 'right': 40, 'top': 60, 'bottom': 60}
            }
        },
        'scales': {
            'xAxes': [{
                'type': "time",
                'time': {
                    'tooltipFormat': 'll',
                    'unit': 'day',
                    'isoWeekday': True,
                    'stepSize': 1,
                    'displayFormats': { 'week': 'ddd D MMM YYYY' },
                    'scaleLabel': { 'display': True,
                                   'labelString': 'Date' }
                }
            },
            {
                'yAxes': [
                    {
                        'scaleLabel': { 'display': True,
                                       'labelString': 'Value' }
                    }
                ]
            }
        }
    }

    mychart = Chart(dataset, 'line', colorscheme=None, options=options)
    display(mychart)

for col in list(df_clai.select_dtypes(exclude=['datetime', 'object']).columns):
    ipychart_line(df_clai, x="Fecha", y=col, title=col)
```



In [58]:

```
options = {
    'title': {
        'display': True,
        '#         'text': '',
        'fontSize': 30,
        'fontStyle': 'bold',
        'layout': {
            'padding': {'left': 40, 'right': 40, 'top': 60, 'bottom': 60}}
    },
    'scales': {
        'xAxes': [
            {'type': "time",
             'time': {
                 'tooltipFormat': 'll',
                 'unit': 'day',
                 'isoWeekday': True,
                 'stepSize': 1,
                 'displayFormats': {'week': 'ddd D MMM YYYY'}}},
            {'scaleLabel': {'display': True,
                           'labelString': 'Date'}}],
        'yAxes': [
            {'scaleLabel': {'display': True,
                           'labelString': 'Value'}}]
    }
}
```

In [59]:

```
# # https://bit.ly/2TiArQt↔
```

In [60]:

```
dataset = {}↔
mychart = Chart(
    dataset, 'line', colorscheme='tableau.ClassicLight10', options=options) # Base type
mychart
```

```
Chart(layout=Layout(align_self='stretch', height='auto'))
```

```
In [61]: from ipychart import ChartDataFrame

# datalabels_arguments = {'display': False, 'pointRadius': 0.1, 'pointHoverRadius': 5}

g_df = ChartDataFrame(df_clai)
g_df.line(x='Fecha', y='TransaccionCuenta',
          dataset_options={
              'fill': False,
              #                 'datalabels': datalabels_arguments,
              'borderWidth': 1,
              'pointRadius': 1,
              'pointHoverRadius': 5,
              #                 'display': True,
          },
          colorscheme='office.Parallax6',
          options={
              'scales': {'xAxes': [{ 'ticks': { 'fontSize': 5, 'fontStyle': 'italic' }}],
                         'yAxes': [{ 'ticks': { 'min': 0, 'max': 2000, 'fontSize': 5, 'fontStyle': 'italic',
                                              'stepSize': 250, 'minRotation': 15, 'padding': 10 }
                                     }]
          }
      )
)

Chart(layout=Layout(align_self='stretch', height='auto'))
```

```
In [65]: metrics_df = df_clai.copy()
metrics_df.reset_index(drop=True, inplace=True)
```

```
In [67]: # metrics_df
```

## 7 Detección de anomalías

```
In [ ]: # Ref: https://towardsdatascience.com/anomaly-detection-with-isolation-forest-visualization-23cd75c281e2
# Ref: https://towardsdatascience.com/anomaly-detection-for-dummies-15f148e559c1
# Ref: https://towardsdatascience.com/anomaly-detection-with-isolation-forest-visualization-23cd75c281e2
# Ref: https://towardsdatascience.com/isolation-forest-with-statistical-rules-4dd27dad2da9
```

```
In [68]: list(metrics_df.select_dtypes(exclude = ['datetime', 'object']).columns)
```

```
Out[68]: ['TransaccionCuenta',
 'TransaccionResp[ms]',
 'Servers',
 'CPU_Int',
 'CPuLot',
 'Int_Car_Util',
 'Alto_Util_Un',
 'Alto_Util_Disco',
 'ID',
 'usuar',
 'coinc',
 'Excp',
 'Promedio_ES',
 'Promedio_Lecturas',
 'Promedio_Grabaciones',
 'Promedio_K_ES',
 'Promedio_utí',
 'Util_alta',
 'Uni_util_alta',
 'Tiempo_Service_Alto',
 'Unit_Service_alta',
 'Espacio_Disco_Utilizado']
```

In [94]:

```
metrics_df.columns
to_model_columns=list(metrics_df.select_dtypes(exclude = ['datetime', 'object']).columns)
# display(to_model_columns)

aux_df = metrics_df.select_dtypes(include = ['datetime', 'object'])

from sklearn.ensemble import IsolationForest as IF

def anomaly_model(df):

    clf_model = IF(n_estimators=1500,
                    max_samples='auto',
                    contamination=0.07,
                    max_features=1.0,
                    bootstrap=False,
                    n_jobs=-1,
                    random_state=40,
                    verbose=0
                   )

    clf_model.fit(metrics_df[to_model_columns])

    pred = clf_model.predict(metrics_df[to_model_columns])

    metrics_df['anomalia']=pred
    outliers = metrics_df.loc[metrics_df['anomalia'] == -1]
    outlier_index=list(outliers.index)
    print("-"*30, "\n 1 : Valores Normales\n-1 : Anomalías\n", "-"*30)
    print(metrics_df['anomalia'].value_counts())
    return df, outlier_index

metrics_df, out_idx = anomaly_model(df=metrics_df)
```

```
-----  
1 : Valores Normales  
-1 : Anomalías  
-----
```

```
1    2362  
-1    178  
Name: anomalia, dtype: int64
```

```
In [95]: print(f"Porcentaje de anomalías en los datos: { round((len(metrics_df.loc[metrics_df['anomalia']==-1])/len(metrics_df))*100,2) }")
```

Porcentaje de anomalías en los datos: 7.01%

```
In [96]: # Ref: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_excel.html
# Ref: https://stackoverflow.com/questions/35940748/use-python-to-launch-excel-file

# Anomaly Report
from decouple import config

ar_path = 'Datasets\data_anomalias.xlsx'

ar_sheet_1 = metrics_df.copy()
ar_sheet_1_name = "data"

ar_sheet_2 = metrics_df.loc[metrics_df['anomalia']==-1].copy()
ar_sheet_2_name = "Anomalias"

with pd.ExcelWriter(ar_path) as writer:
    ar_sheet_1.to_excel(writer, index=False, sheet_name=ar_sheet_1_name)
    ar_sheet_2.to_excel(writer, index=False, sheet_name=ar_sheet_2_name)

# Abrir Archivo de Anomalias
# open_file = True
open_file = config('OPEN_FILE', default='False', cast=bool)
if open_file:
    os.system(f"start EXCEL.EXE {ar_path}")
```

```
In [97]: # metrics_df.shape
```

```
aux_df
```

Out[97]:

|      | Fecha               |
|------|---------------------|
| 0    | 2021-03-12 00:55:00 |
| 1    | 2021-03-12 01:00:00 |
| 2    | 2021-03-12 01:10:00 |
| 3    | 2021-03-12 01:15:00 |
| 4    | 2021-03-12 01:20:00 |
| ...  | ...                 |
| 2535 | 2021-03-22 09:25:15 |
| 2536 | 2021-03-22 09:30:15 |
| 2537 | 2021-03-22 09:35:15 |
| 2538 | 2021-03-22 09:40:15 |
| 2539 | 2021-03-22 09:45:15 |

2540 rows × 1 columns

```
In [98]: from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from mpl_toolkits.mplot3d import Axes3D

pca = PCA(n_components=3) # k=3
scaler = StandardScaler()
X = scaler.fit_transform(metrics_df[to_model_columns])
X_pca = pca.fit_transform(X)
```

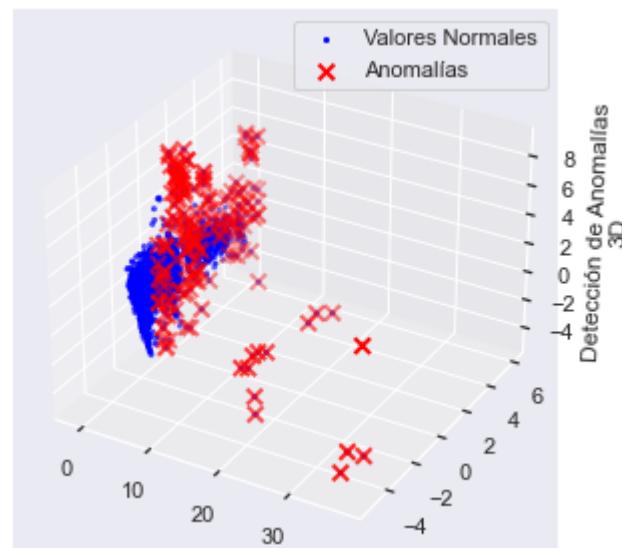
In [99]:

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_zlabel("Deteción de Anomalías\n3D")

ax.scatter(X_pca[:, 0],
           X_pca[:, 1],
           zs=X_pca[:, 2],
           s=4, lw=1,
           label="Valores Normales",
           c="blue"
          )

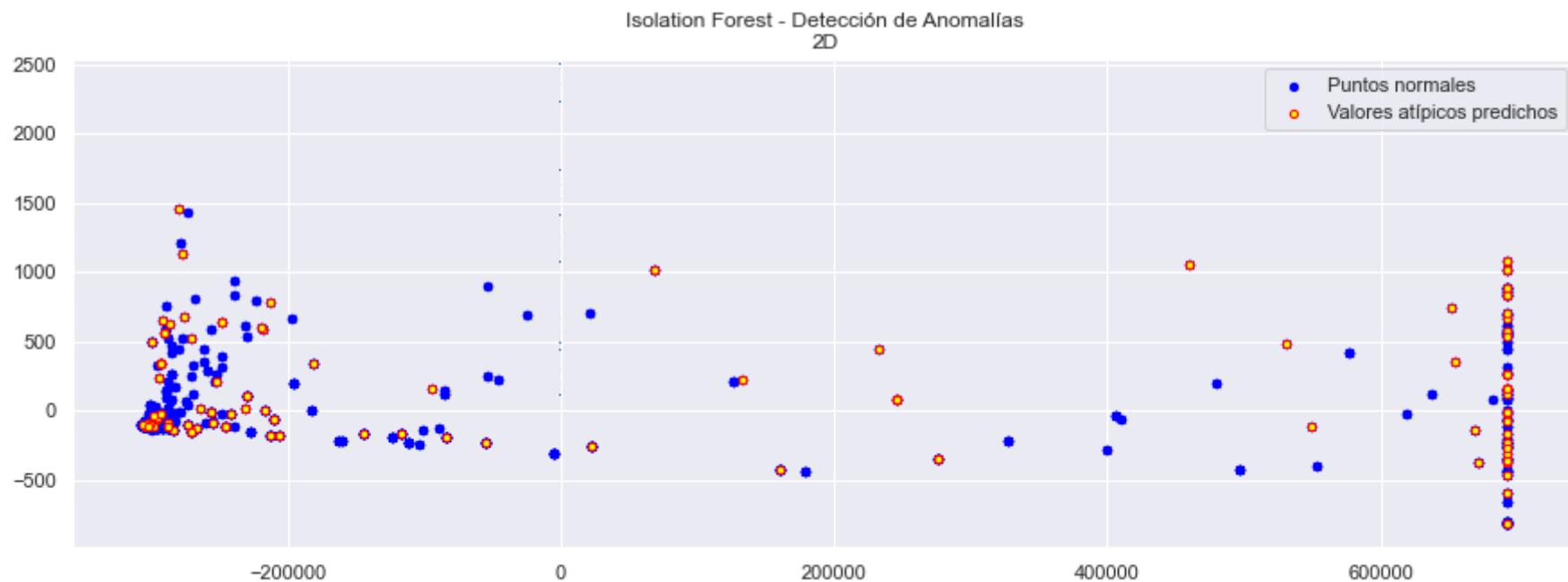
ax.scatter(X_pca[out_idx, 0],
           X_pca[out_idx, 1],
           X_pca[out_idx, 2],
           lw=2,
           s=60,
           marker="x",
           c="red",
           label="Anomalías"
          )

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



In [113]:

```
from sklearn.decomposition import PCA
pca = PCA(2)
pca.fit(metrics_df[to_model_columns])
rto = pd.DataFrame(pca.transform(metrics_df[to_model_columns]))
Z = np.array(rto)
plt.title("Isolation Forest - Detección de Anomalías\n2D")
plt.contourf(Z, cmap=plt.cm.Blues_r)
b1 = plt.scatter(rto[0], rto[1], c='blue', s=18, label="Puntos normales")
b1 = plt.scatter(rto.iloc[out_idx, 0], rto.iloc[out_idx, 1], c='yellow', s=18, edgecolor="red", label="Valores atípicos predichos")
plt.legend(loc="upper right")
plt.show()
```



In [ ]: # metrics\_df

```
In [114]: metrics_df['Fecha'] = aux_df['Fecha']
# metrics_df
```

```
In [115]: CodName = {}
clsName = list(metrics_df.columns)
for cod in range(len(clsName)):
    CodName[clsName[cod]] = cod
print(CodName)
```

```
{'Fecha': 0, 'TransaccionCuenta': 1, 'TransaccionResp[ms]': 2, 'Servers': 3, 'CPU_Int': 4, 'CPuLot': 5, 'Int_Car_Util': 6, 'Alto_Util_Un': 7, 'Alto_Util_Disco': 8, 'ID': 9, 'usuari': 10, 'coinc': 11, 'Excp': 12, 'Promedio_ES': 13, 'Promedio_Lecturas': 14, 'Promedio_Grabaciones': 15, 'Promedio_K_ES': 16, 'Promedio_utii': 17, 'Util_alta': 18, 'Uni_util_alta': 19, 'Tiempo_Service_Alto': 20, 'Unit_Service_alta': 21, 'Espacio_Disco_Utilizado': 22, 'anomalia': 23}
```

```
In [116]: n1=0
n2=14

col_sel = [ metrics_df.columns[n1], metrics_df.columns[n2] ]
col_sel
```

```
Out[116]: ['Fecha', 'Promedio_Lecturas']
```

```
In [117]: df = pd.DataFrame([])
df = metrics_df.copy()
```

In [118]:

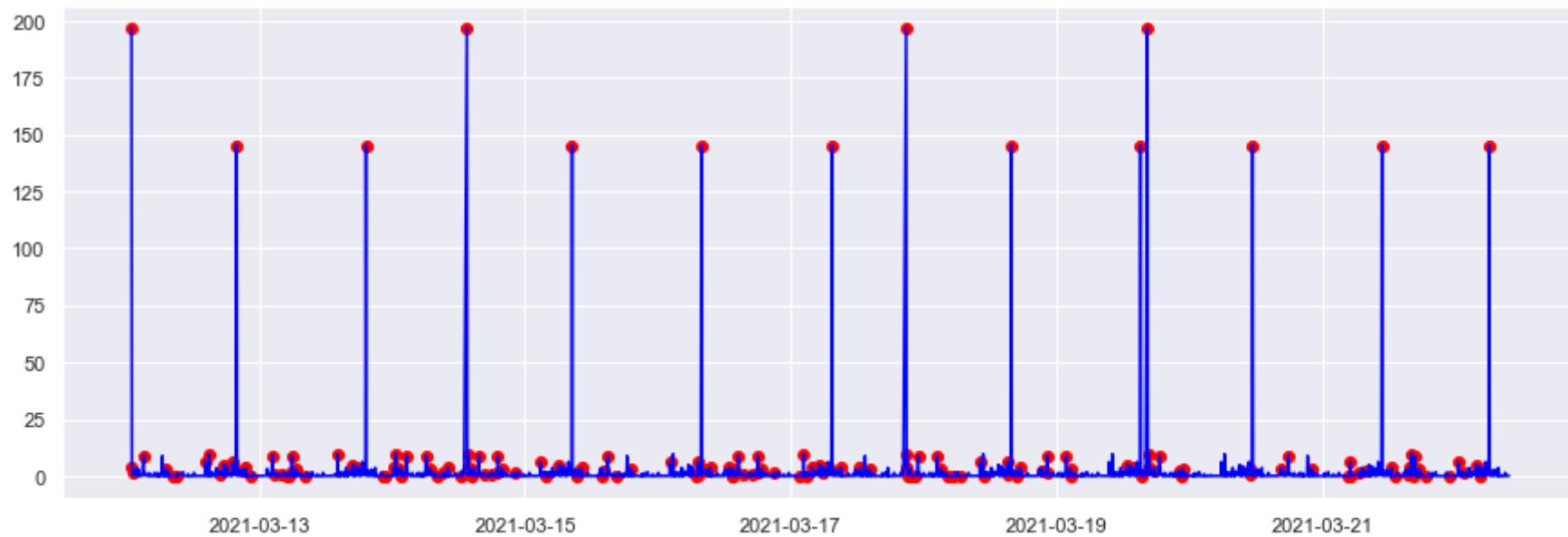
```
# visualisation of anomaly throughout time (viz 1)
fig, ax = plt.subplots()

col_sel = [ df.columns[n1], df.columns[n2] ]

# a = df.loc[df['anomaly'] == -1, ['time_epoch', 'value']] #anomaly
a = df.loc[df['anomalia'] == -1, col_sel] #anomaly

print(col_sel[0])
ax.plot(df[col_sel[0]], df[col_sel[1]], color='blue')
ax.scatter(a[col_sel[0]], a[col_sel[1]], color='red')
plt.show()
```

Fecha



In [119]:

```
display(df[['Fecha', 'TransaccionCuenta', 'anomalia']].query('anomalia == -1').sample(5))
display(df[['Fecha', 'TransaccionCuenta', 'anomalia']].query('anomalia == 1').sample(5))
```

|      | Fecha               | TransaccionCuenta | anomalia |
|------|---------------------|-------------------|----------|
| 1836 | 2021-03-19 13:30:11 | 463               | -1       |
| 1265 | 2021-03-17 05:40:08 | 585               | -1       |
| 1335 | 2021-03-17 12:30:08 | 5                 | -1       |
| 296  | 2021-03-13 06:35:02 | 471               | -1       |
| 200  | 2021-03-12 21:00:01 | 355               | -1       |

|      | Fecha               | TransaccionCuenta | anomalia |
|------|---------------------|-------------------|----------|
| 2441 | 2021-03-22 00:20:14 | 579               | 1        |
| 1110 | 2021-03-16 14:10:07 | 516               | 1        |
| 979  | 2021-03-16 01:25:06 | 0                 | 1        |
| 547  | 2021-03-14 07:30:03 | 1255              | 1        |
| 236  | 2021-03-13 00:25:01 | 17                | 1        |

In [120]:

```
display(df[['Fecha', 'Promedio_ES', 'anomalia']].query('anomalia == -1').sample(5))
display(df[['Fecha', 'Promedio_ES', 'anomalia']].query('anomalia == 1').sample(5))
```

|      | Fecha               | Promedio_ES | anomalia |
|------|---------------------|-------------|----------|
| 1278 | 2021-03-17 07:00:08 | 19.6        | -1       |
| 554  | 2021-03-14 08:05:03 | 16.0        | -1       |
| 1244 | 2021-03-17 03:50:07 | 6.1         | -1       |
| 2357 | 2021-03-21 16:20:14 | 17.2        | -1       |
| 1555 | 2021-03-18 10:55:09 | 1.7         | -1       |

|      | Fecha               | Promedio_ES | anomalia |
|------|---------------------|-------------|----------|
| 100  | 2021-03-12 11:10:01 | 3.5         | 1        |
| 340  | 2021-03-13 10:40:02 | 1.4         | 1        |
| 972  | 2021-03-16 00:45:06 | 1.8         | 1        |
| 1562 | 2021-03-18 11:30:09 | 6.1         | 1        |
| 1745 | 2021-03-19 04:55:10 | 1.3         | 1        |

In [121]:

```
# visualisation of anomaly throughout time (viz 1)
fig, ax = plt.subplots()

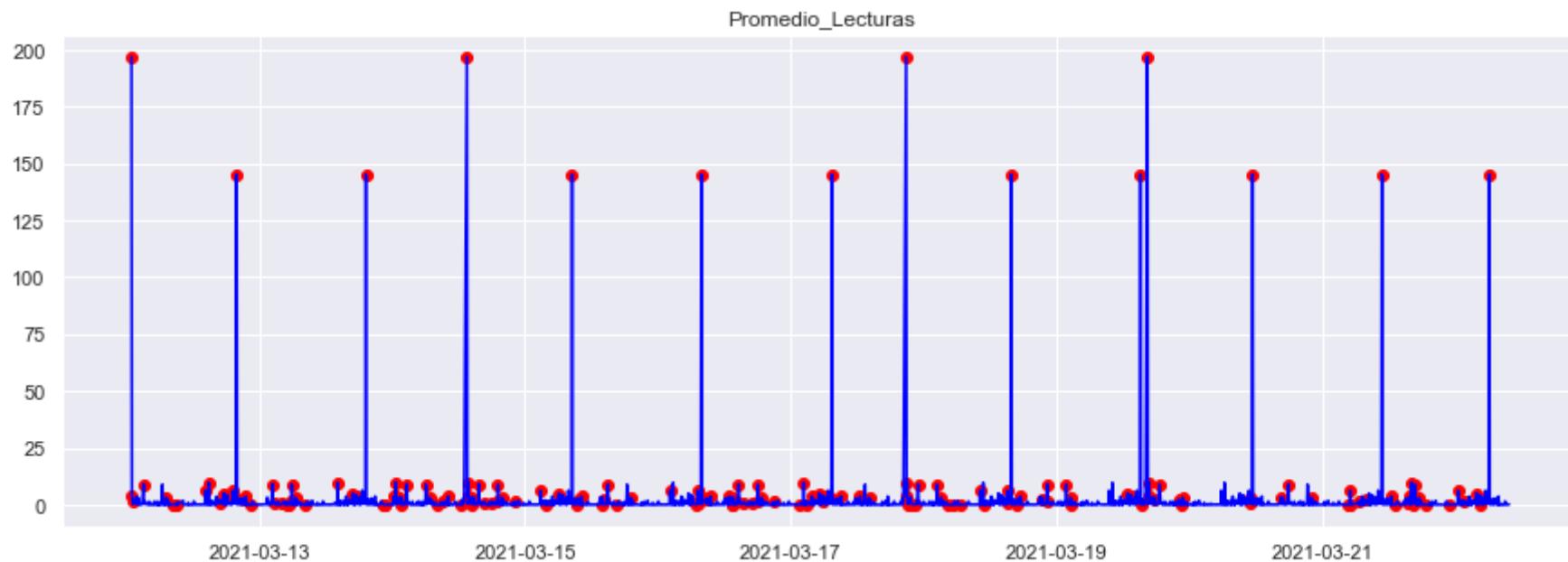
col_sel = [ df.columns[n1], df.columns[n2] ]

# a = df.loc[df['anomaly'] == -1, ['time_epoch', 'value']] #anomaly
a = df.loc[df['anomalia'] == -1, col_sel] #anomaly

print(col_sel[1])
plt.title(col_sel[1])

ax.plot(df[col_sel[0]], df[col_sel[1]], color='blue')
ax.scatter(a[col_sel[0]],a[col_sel[1]], color='red')
plt.show()
```

Promedio\_Lecturas



In [122]:

a

Out[122]:

|      | Fecha               | Promedio_Lecturas |
|------|---------------------|-------------------|
| 0    | 2021-03-12 00:55:00 | 196.8             |
| 1    | 2021-03-12 01:00:00 | 3.8               |
| 3    | 2021-03-12 01:15:00 | 1.2               |
| 19   | 2021-03-12 03:05:00 | 9.1               |
| 66   | 2021-03-12 07:15:00 | 3.2               |
| ...  | ...                 | ...               |
| 2443 | 2021-03-22 00:30:14 | 6.3               |
| 2450 | 2021-03-22 01:30:14 | 1.8               |
| 2475 | 2021-03-22 03:50:15 | 5.0               |
| 2483 | 2021-03-22 04:40:15 | 0.2               |
| 2501 | 2021-03-22 06:10:15 | 145.1             |

178 rows × 2 columns

In [123]: df.head(4)

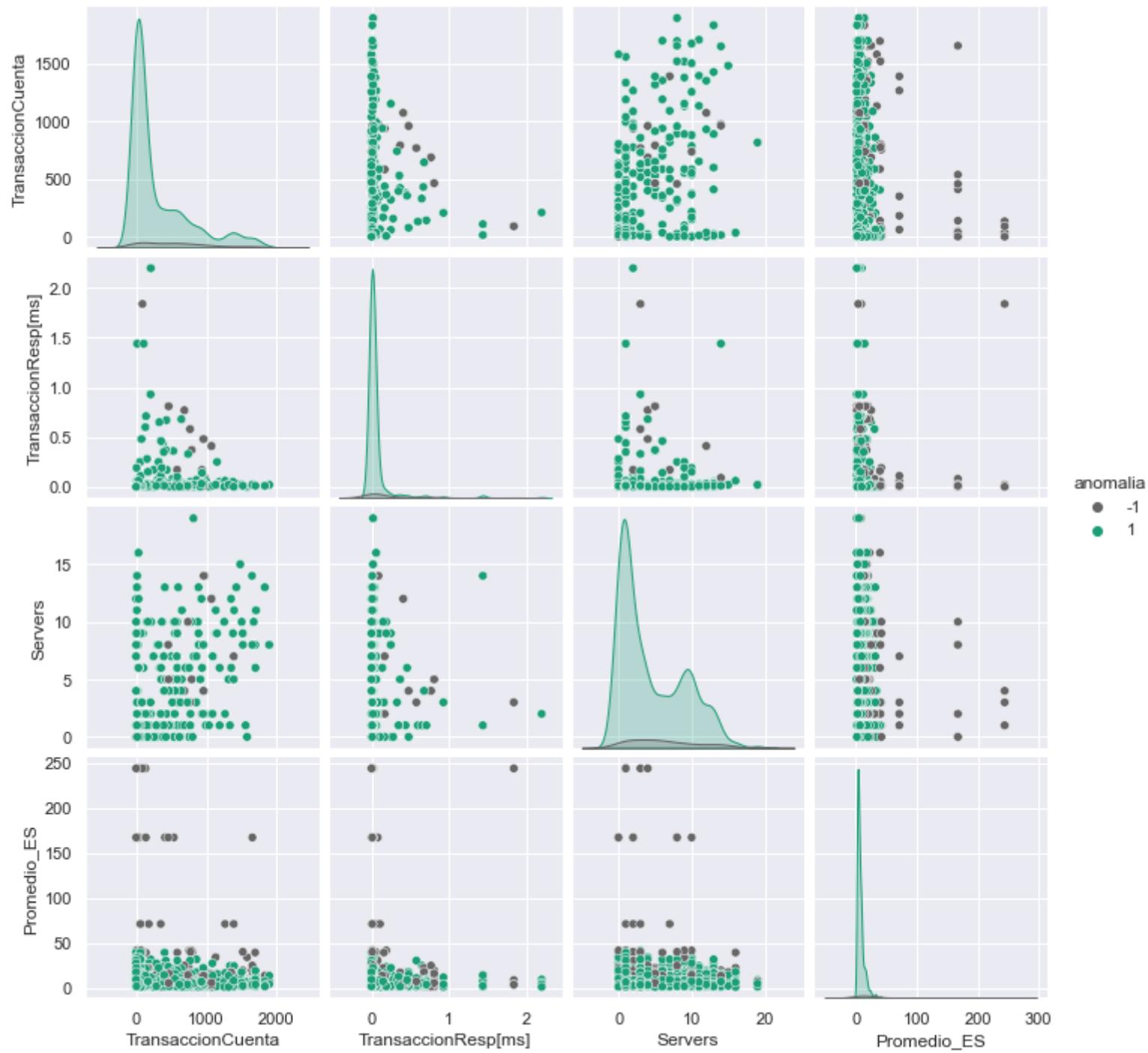
Out[123]:

|   | Fecha               | TransaccionCuenta | TransaccionResp[ms] | Servers | CPU_Int | CPuLot | Int_Car_Util | Alto_Util_Un | Alto_Util_Disco | ID | usuар | coinc | Excp   | Promedio |
|---|---------------------|-------------------|---------------------|---------|---------|--------|--------------|--------------|-----------------|----|-------|-------|--------|----------|
| 0 | 2021-03-12 00:55:00 | 36                | 0.02                | 1       | 0       | 1      | 0            | 4            | 156             | 0  | 133   | 2     | 20084  | 2        |
| 1 | 2021-03-12 01:00:00 | 0                 | 0.00                | 1       | 0       | 1      | 0            | 4            | 156             | 0  | 171   | 2     | 94267  |          |
| 2 | 2021-03-12 01:10:00 | 18                | 0.00                | 3       | 0       | 3      | 0            | 5            | 150             | 0  | 310   | 2     | 999999 |          |
| 3 | 2021-03-12 01:15:00 | 0                 | 0.00                | 4       | 0       | 4      | 0            | 6            | 156             | 0  | 250   | 2     | 999999 |          |

In [124]:

```
sns.color_palette("mako", as_cmap=True)
g= sns.pairplot(df[['TransaccionCuenta', 'TransaccionResp[ms]', "Servers", "Promedio_ES", "anomalia']], hue="anomalia", palette='Dark2_r')
g.fig.suptitle("Pairplot", y=1.02)
plt.show()
```

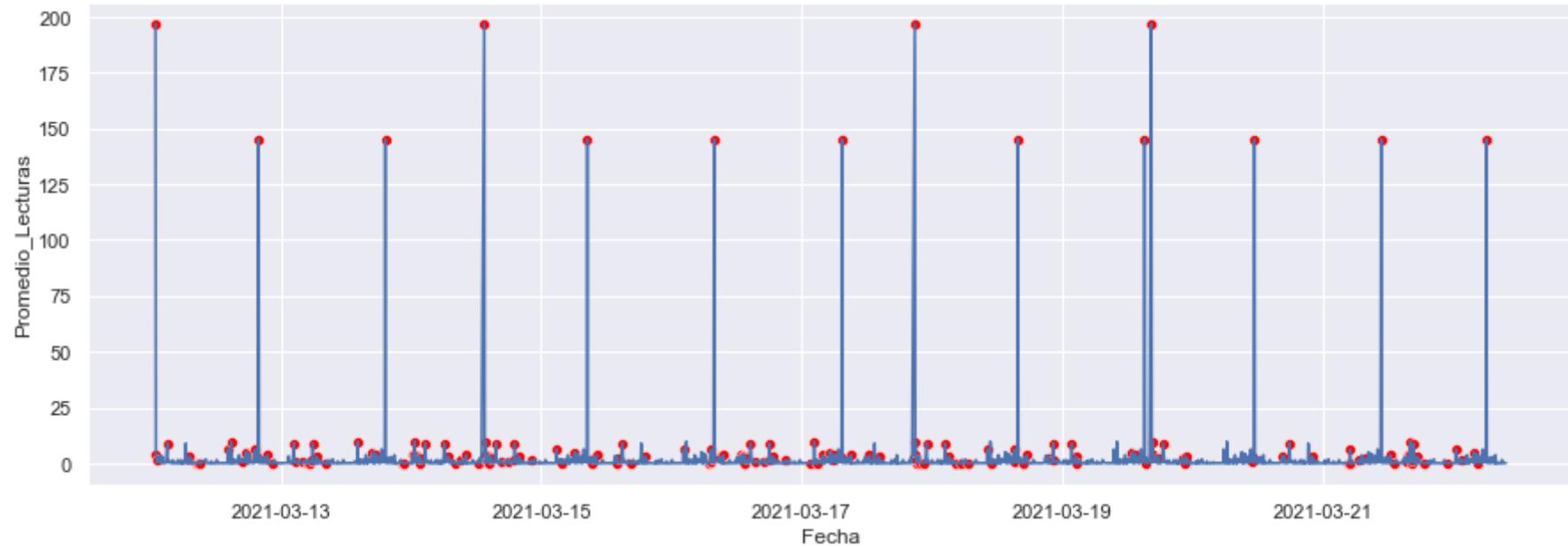
Pairplot



In [127]:

```
fig, ax = plt.subplots()
sns.lineplot(df[col_sel[0]], df[col_sel[1]], ax=ax)
sns.scatterplot(a[col_sel[0]], a[col_sel[1]], color='red', ax=ax)
```

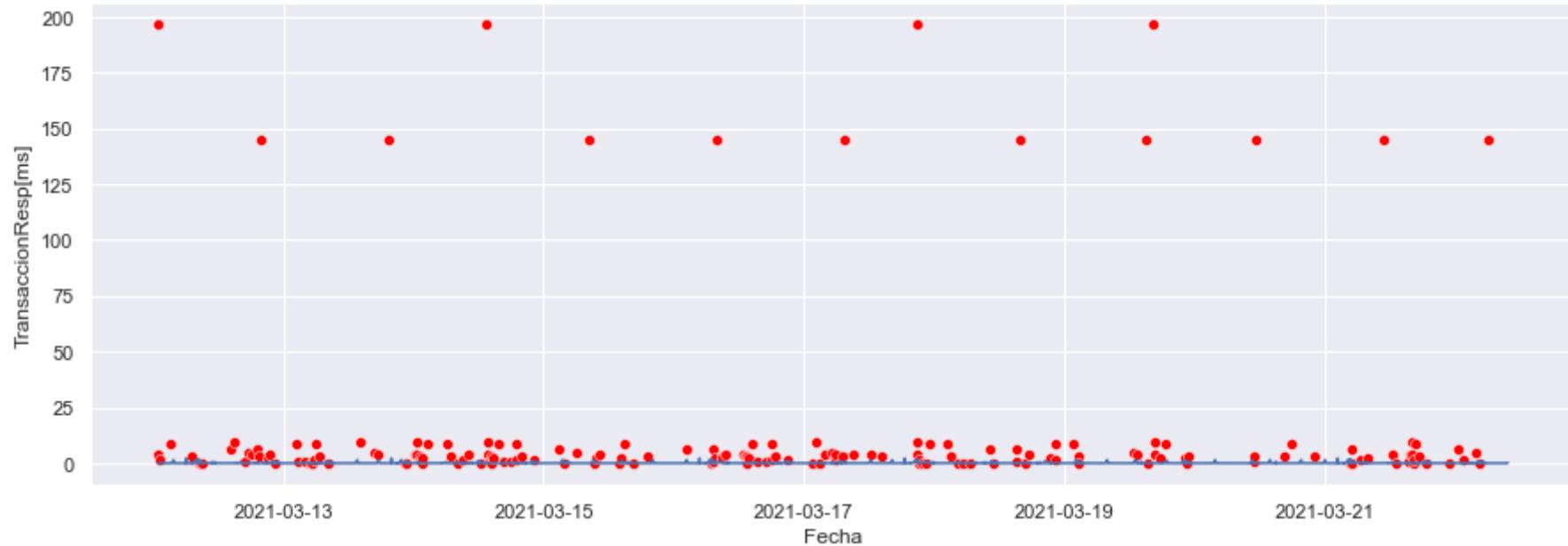
Out[127]: <AxesSubplot:xlabel='Fecha', ylabel='Promedio\_Lecturas'>



In [128]:

```
fig, ax = plt.subplots()  
sns.lineplot(data=df, x=df["Fecha"],  
              y=df["TransaccionResp[ms]"],  
              # hue="Estado_trx",  
              ax=ax)  
sns.scatterplot(a[col_sel[0]], a[col_sel[1]], color='red', ax=ax)
```

Out[128]: <AxesSubplot:xlabel='Fecha', ylabel='TransaccionResp[ms]'>



In [129]:

```
fig, ax = plt.subplots()

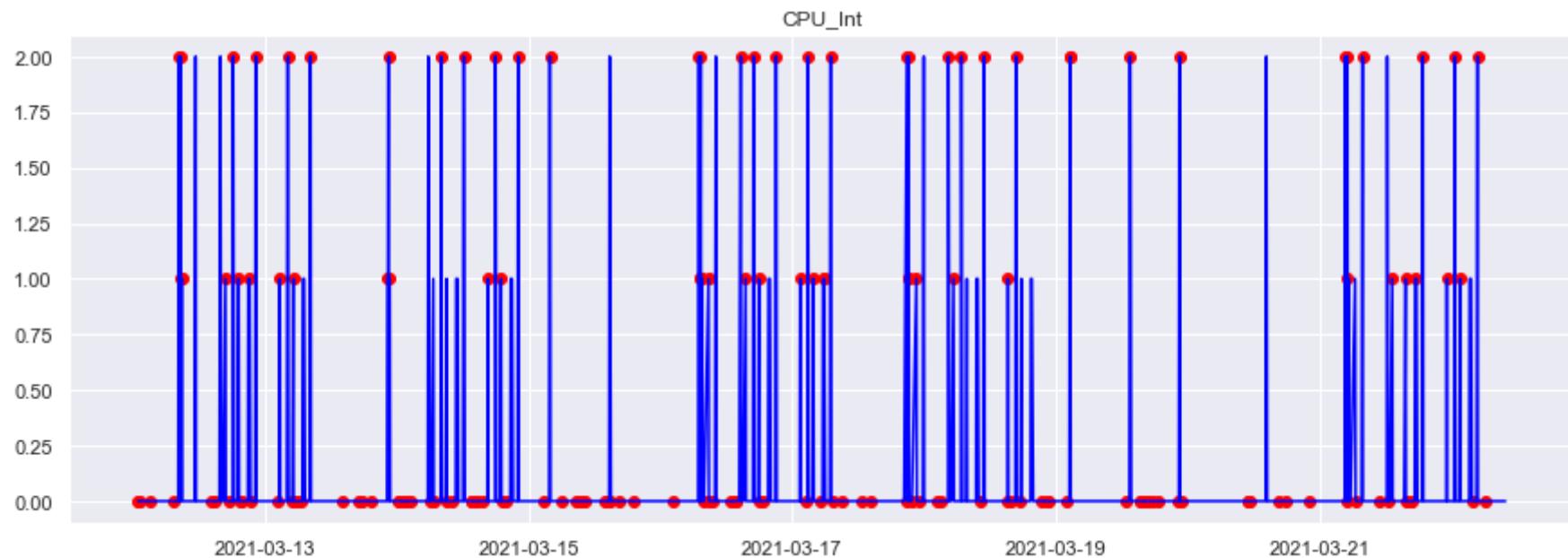
col_sel = [ df.columns[n1], df.columns[4] ]

# a = df.loc[df['anomaly'] == -1, ['time_epoch', 'value']] #anomaly
a = df.loc[df['anomalia'] == -1, col_sel] #anomaly

print(col_sel[1])
plt.title(col_sel[1])

ax.plot(df[col_sel[0]], df[col_sel[1]], color='blue')
ax.scatter(a[col_sel[0]],a[col_sel[1]], color='red')
plt.show()
```

CPU\_Int



```
In [130]: CodName
```

```
Out[130]: {'Fecha': 0,
 'TransaccionCuenta': 1,
 'TransaccionResp[ms]': 2,
 'Servers': 3,
 'CPU_Int': 4,
 'CPuLot': 5,
 'Int_Car_Util': 6,
 'Alto_Util_Un': 7,
 'Alto_Util_Disco': 8,
 'ID': 9,
 'usuar': 10,
 'coinc': 11,
 'Excp': 12,
 'Promedio_ES': 13,
 'Promedio_Lecturas': 14,
 'Promedio_Grabaciones': 15,
 'Promedio_K_ES': 16,
 'Promedio_utí': 17,
 'Util_alta': 18,
 'Uni_util_alta': 19,
 'Tiempo_Service_Alto': 20,
 'Unit_Service_alta': 21,
 'Espacio_Disco_Utilizado': 22,
 'anomalia': 23}
```

```
In [131]: df.TransaccionCuenta.describe()
```

```
Out[131]: count    2540.000
mean      328.841
std       457.478
min       0.000
25%      12.000
50%      78.000
75%     549.000
max     1895.000
Name: TransaccionCuenta, dtype: float64
```

In [132]:

```
# Datos extraños
df["Estado_trx"] = (
    np.select(
        condlist=[
            df['TransaccionCuenta'] <= 12,
            df['TransaccionCuenta'] >= 700,
        ],
        choicelist=[
            -1,
            1
        ],
        default= 0
))
```

Desbalanceado

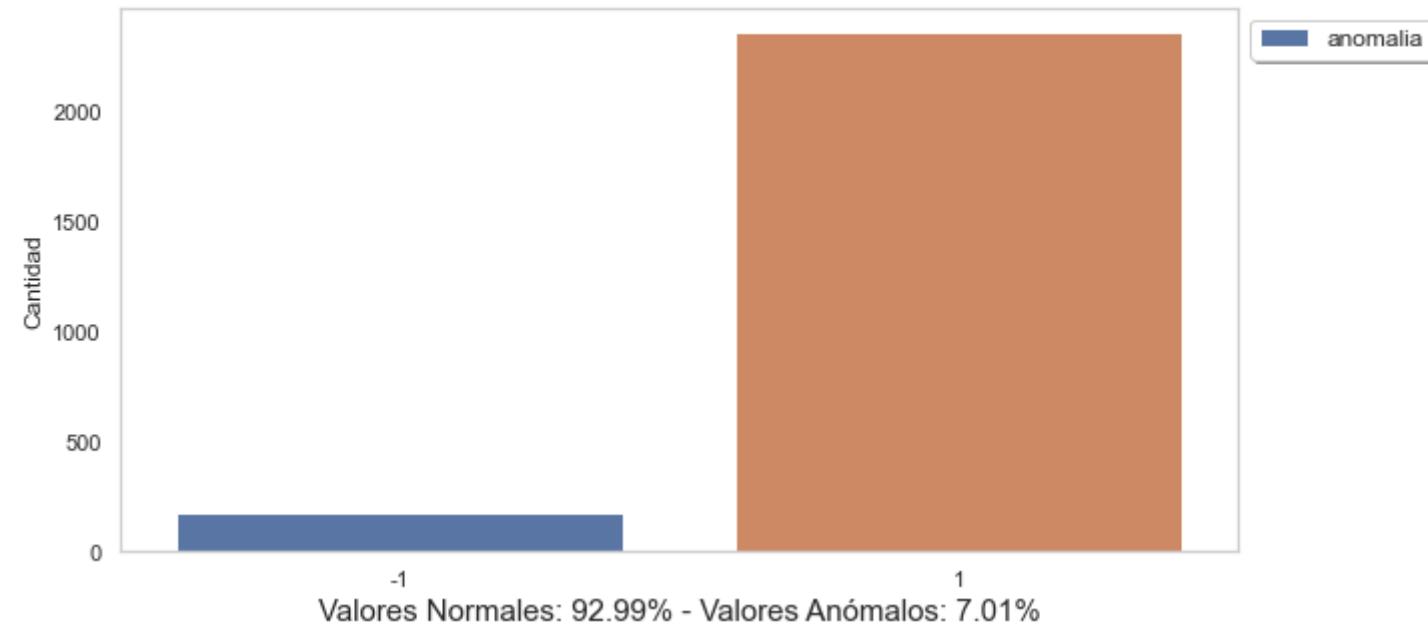
In [133]:

```
Normales, Anomalias= df.anomalia.value_counts()
cnt_anomalia = round(Anomalias/len(df) * 100, 2)
cnt_normales = round(Normales/len(df) * 100, 2)

# print("Valores Normales: ", cnt_normales, '% del dataset')
# print("Valores Anomalos: ", cnt_anomalia, '% del dataset')

sns.set_theme(style="whitegrid")
fg, ax = plt.subplots(figsize=(10, 5))
ax.grid()
ax = sns.countplot(x="anomalia", data=df, label="anomalia")
ax.set_title("Valores Anómalos vs. Valores Normales", fontsize = 18, y =1.05)
ax.set_xlabel(f"Valores Normales: {cnt_normales}% - Valores Anómalos: {cnt_anomalia}%", fontsize = 15, y =1.05)
ax.set_ylabel("Cantidad")
# ax.legend(loc="best")
ax.legend(fontsize = 12,
          bbox_to_anchor= (1, 1),
#           title="Colores",
          title_fontsize=14,
          shadow=True,
          facecolor='white')
plt.savefig(f"img/testpng.png", format='png', dpi=250)
plt.show()
```

## Valores Anómalos vs. Valores Normales



In [134]:

```
# https://seaborn.pydata.org/examples/many_pairwise_correlations.html?highlight=corr

# Compute the correlation matrix
corr = df_clai.select_dtypes(exclude=['datetime', 'object']).corr()

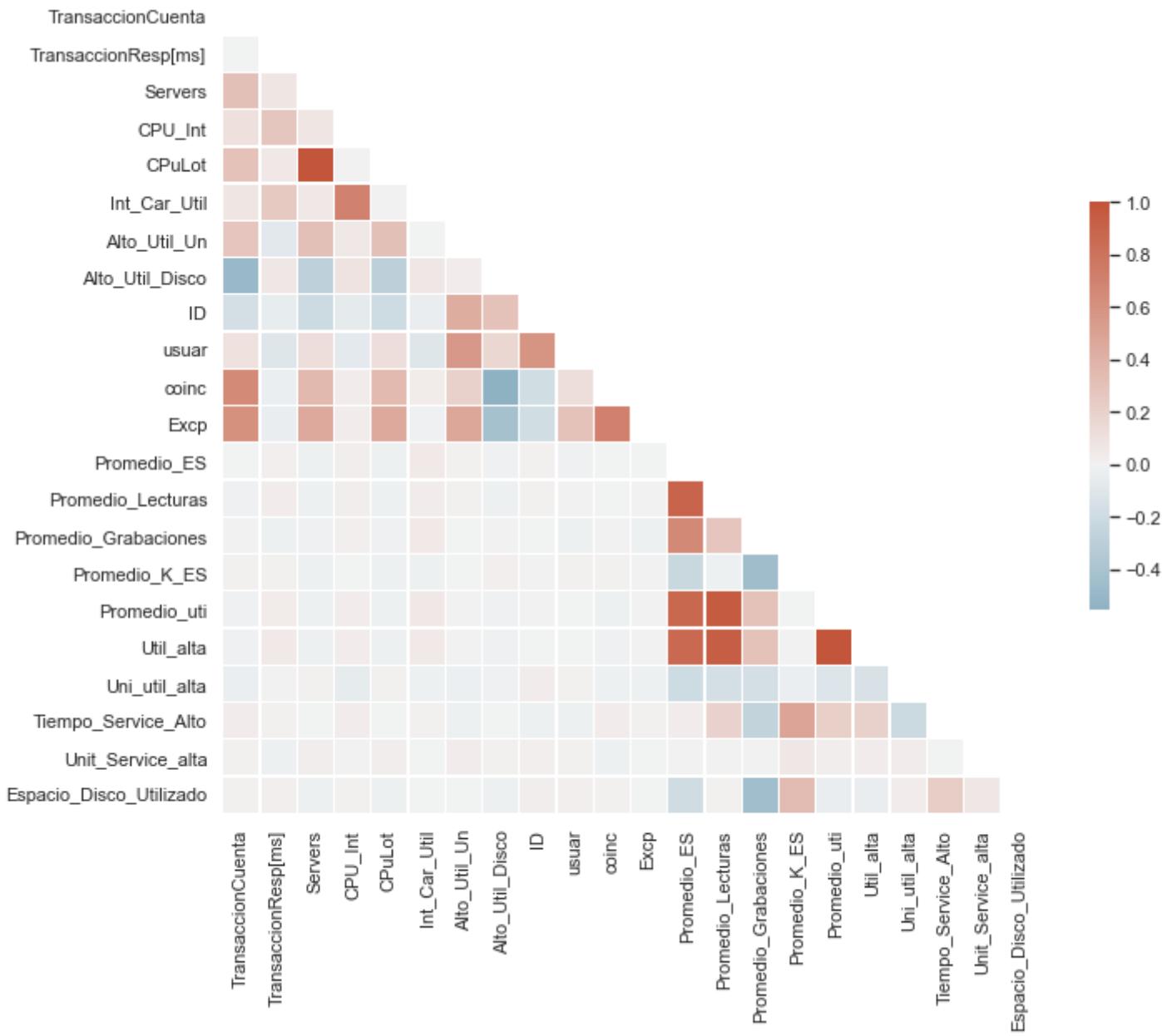
# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=1, center=0,
            square=True, linewidths=.3, cbar_kws={"shrink": .5})
```

Out[134]: <AxesSubplot:>



## 8 Predicción de anomalías

### 8.1 Modulos - sklearn

In [135]:

```
from sklearn import preprocessing
from sklearn import svm
from sklearn.ensemble import AdaBoostClassifier, ExtraTreesClassifier
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import learning_curve
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.tree import plot_tree
import scikitplot as skplt
```

In [137]:

```
# La semilla permite obtener repetibilidad en los resultados de los algoritmos.
semilla_aleatoria = 2021
```

In [138]:

```
# Se seleccionan las columnas para el proceso predictivo.
columnas_x=df.columns.difference(['anomalia','Fecha', 'Estado_trx'])

# Variables
X=df[columnas_x]

# Clase objetivo
Y=df['anomalia']
```

In [139]:

```
# list(columnas_x)
```

## 8.2 Modelos

### LinearRegression

- [https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LinearRegression.html?highlight=linearregression#sklearn.linear\\_model.LinearRegression](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html?highlight=linearregression#sklearn.linear_model.LinearRegression) ([https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LinearRegression.html?highlight=linearregression#sklearn.linear\\_model.LinearRegression](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html?highlight=linearregression#sklearn.linear_model.LinearRegression))

### DecisionTreeClassifier

- <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decisiontreeclassifier#sklearn.tree.DecisionTreeClassifier> (<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decisiontreeclassifier#sklearn.tree.DecisionTreeClassifier>)

### ExtraTreesClassifier

Metaestimador que se ajusta a una serie de árboles de decisión aleatorios (también conocidos como árboles extra) en varias submuestras del conjunto de datos y usa el promedio para mejorar la precisión predictiva y controlar el sobreajuste.

- <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html?highlight=decisiontreeregressor#sklearn.tree.DecisionTreeRegressor> (<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html?highlight=decisiontreeregressor#sklearn.tree.DecisionTreeRegressor>)
- <https://www.kite.com/python/docs/sklearn.ensemble.ExtraTreesClassifier> (<https://www.kite.com/python/docs/sklearn.ensemble.ExtraTreesClassifier>)

### RandomForestClassifier & RandomForestRegressor

Un bosque aleatorio es un metaestimador que se ajusta a una serie de clasificadores de árboles de decisión en varias submuestras del conjunto de datos y utiliza promedios para mejorar la precisión predictiva y controlar el sobreajuste. El tamaño de la submuestra se controla con el parámetro max\_samples si bootstrap = True (predeterminado); de lo contrario, se usa todo el conjunto de datos para construir cada árbol.

- <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=randomforestclassifier#sklearn.ensemble.RandomForestClassifier> (<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=randomforestclassifier#sklearn.ensemble.RandomForestClassifier>)
- <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html?highlight=randomforestregressor#sklearn.ensemble.RandomForestRegressor> (<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html?highlight=randomforestregressor#sklearn.ensemble.RandomForestRegressor>)

In [140]:

```
In [141]: models_dic = {}
for nm, md in models_clf:
    models_dic[nm] = str(md)[0:str(md).find("(")]

for nm, md in models_reg:
    models_dic[nm] = str(md)[0:str(md).find("(")]

models_dic
```

```
Out[141]: {'CART': 'DecisionTreeClassifier',
'ECART': 'ExtraTreesClassifier',
'RFC': 'RandomForestClassifier',
'LR': 'LinearRegression',
'LogR': 'LogisticRegression',
'CART_Reg': 'DecisionTreeRegressor',
'RFR': 'RandomForestRegressor'}
```

## 8.3 Evaluación de algoritmos de aprendizaje automático

```
In [142]: # Dividimos el dataset en 80% de datos para entrenar y 20% para test
train_x, test_x, train_y, test_y = train_test_split(X,
                                                    Y,
                                                    test_size=0.2,
                                                    random_state=semilla_aleatoria
                                                   )
```

### 8.3.1 Validación cruzada

La validación cruzada (**cross-validation**) es un método estadístico que se utiliza para estimar la capacidad de clasificación de los modelos de aprendizaje automático. También, se puede definir como un método que permite estimar el ajuste del modelo a un hipotético conjunto de datos de prueba cuando no se dispone de este conjunto de datos de prueba de manera explícita.

Consiste en dividir el conjunto de ejemplos disponibles en un conjunto de datos de entrenamiento y un conjunto de datos de validación:

- **Datos de entrenamiento:** se utilizan para generar el árbol.
- **Datos de validación:** se utilizan para validar la precisión del árbol generado sobre datos futuros.

Ref:

- <https://machinelearningmastery.com/k-fold-cross-validation/> (<https://machinelearningmastery.com/k-fold-cross-validation/>)
- <https://machinelearningmastery.com/how-to-configure-k-fold-cross-validation/> (<https://machinelearningmastery.com/how-to-configure-k-fold-cross-validation/>)

In [143]:

```
#-----#
# Configurar el método de remuestreo
#-----#
nparts=10

# KFold
# KFold is a cross-validator that divides the dataset into k folds.
kfold = KFold(n_splits=nparts, shuffle=True, random_state=semilla_aleatoria)

# StratifiedKFold
# Stratified is to ensure that each fold of dataset has the same proportion of observations with a given label.
kfold_s = StratifiedKFold(n_splits=nparts, random_state=None, shuffle=False)

# RepeatedStratifiedKFold
# As the name depicts this method repeats Stratified K-Fold Cross validator n times with different randomisation
# in each repetition. Same as Stratified K-Fold entire data set is divided into k subsets.
# For each iteration it approximately maintains same percentage of samples of each target class
kfold_Rs = RepeatedStratifiedKFold(n_splits=nparts, n_repeats=3, random_state=1)

# Cargamos los algoritmos
cvs = []

# List métodos de remuestreo
cvs.append(("KFold", kfold))
cvs.append(("StratifiedKFold", kfold_s))
cvs.append(("RepeatedStratifiedKFold", kfold_Rs))
# display(cvs)

# El parametro cv determina la estrategia de división de validación cruzada.

# Ref:
# https://stackoverflow.com/questions/65318931/stratifiedkfolds-vs-kfolds-in-scikit-learn
# https://medium.com/@venkatasujit272/overview-of-cross-validation-3785d5414ece
```

In [144]:

```
def graficar_r2_scores(estimator, train_x, train_y, test_x, test_y, jobs=None, name=None, cv_sel=kfold):
    # Config Gerafica
    fig,axes = plt.subplots(figsize=(7, 3))
    axes.set_title(f"{name}\nR2/Nro. Fold")
    axes.set_xlabel("Nro. Fold\nNúmero de interaciones")
    axes.set_ylabel("R2")
    # Evaluar una puntuación mediante validación cruzada
    train_scores = cross_val_score(estimator, train_x, train_y, cv = cv_sel, n_jobs=jobs,scoring="r2")
    test_scores = cross_val_score(estimator, test_x, test_y, cv = cv_sel, n_jobs=jobs,scoring="r2")
    train_sizes = range(1, nparts+1, 1)
    axes.grid()
    axes.plot(train_sizes, train_scores, 'o-', color="r",label="Datos Entrenamiento")
    axes.plot(train_sizes, test_scores, 'o-', color="b",label="Validacion Cruzada")
    axes.legend(loc="best")

    return train_scores
```

## 8.4 Tamaño de los conjuntos de datos (Entrenamiento y Pruebas)

In [145]:

```
print("Datos de entrenamiento:", sum(list(train_x.count())))
print("Datos de validación: ", sum(list(test_x.count())))

dtrain= {k: list(train_y).count(k) for k in np.unique(train_y) }
dtest= {k: list(test_y).count(k) for k in np.unique(test_y) }

print("\nDatos de entrenamiento:", dtrain)
print("Datos de validación: ", dtest)
```

Datos de entrenamiento: 44704

Datos de validación: 11176

Datos de entrenamiento: {-1: 133, 1: 1899}

Datos de validación: {-1: 45, 1: 463}

## 8.5 Seleccion del método de remuestreo

```
In [146]: cv_sel = kfold_s  
jobs =2
```

```
In [147]: model_resul = {'ModelName': [],  
                      'ModelCode': [],  
                      'ModelType':[],  
                      'Scoring':[],  
                      'ModelScore': [],  
                      'ModelSTD': []}  
results = []  
names = []  
pred_models = []
```

## 8.6 Modelos de regresión

```
In [148]: # Resultados cross_val_score / scoring="r2"

for name, model in models_reg:

    reg_clf_model = model
    reg_clf_model.fit(train_x, train_y)
    pred_model = reg_clf_model.predict(test_x)

    #     # Predicciones
    #     pred_models.append((name, pred_model))

    print('%s: % (name))
    print('Error absoluto medio - MAE: ',
          mean_absolute_error(test_y, pred_model))
    print('Error cuadrático medio - MSE: ',
          mean_squared_error(test_y, pred_model))
    print('Coeficiente de determinación - R2: ', r2_score(test_y, pred_model))
    print("\n")

    graficar_r2_scores(reg_clf_model,
                        train_x,
                        train_y,
                        test_x,
                        test_y,
                        jobs=2,
                        name=name,
                        cv_sel=kfold_s
                      )
    print("\n")

# https://ichi.pro/es/interpretacion-del-error-cuadratico-medio-de-un-modelo-de-regresion-lineal-89502398877112
# https://sitiobigdata.com/2018/08/27/machine-learning-metricas-regresion-mse/#
```

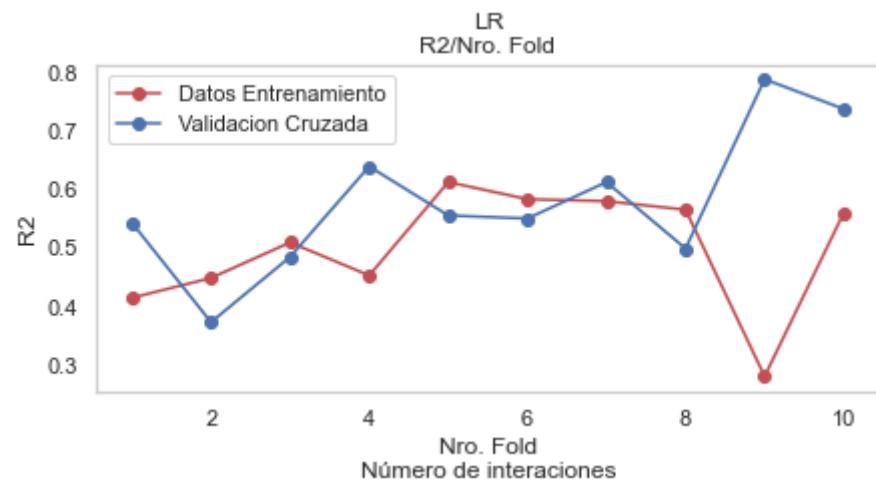
LR:  
Error absoluto medio - MAE: 0.20501422645095613  
Error cuadrático medio - MSE: 0.147587955028497  
Coeficiente de determinación - R2: 0.5429909044099586

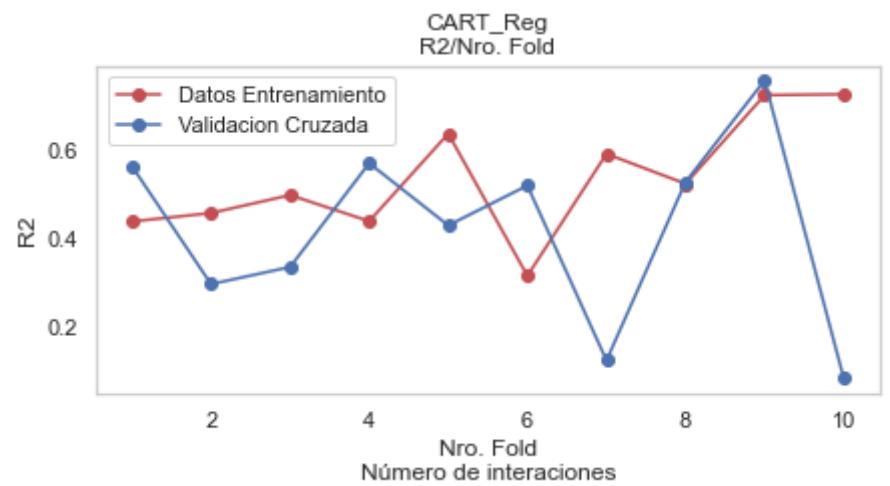
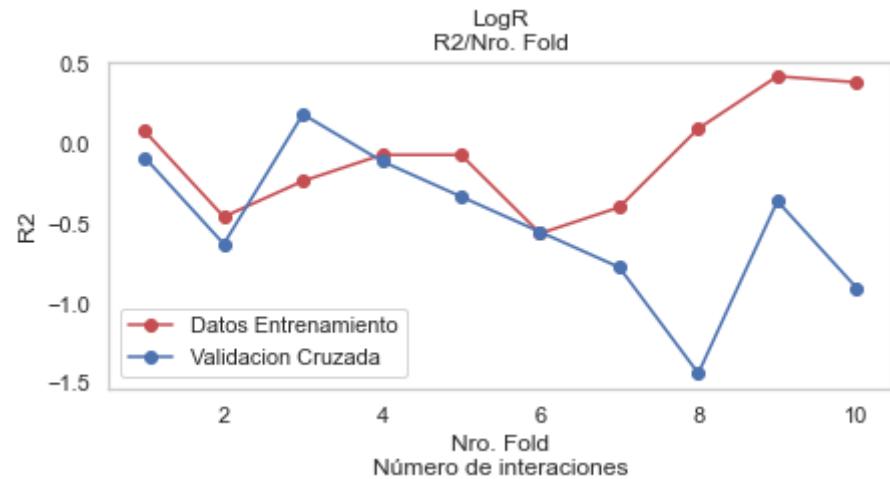
LogR:

Error absoluto medio - MAE: 0.12992125984251968  
Error cuadrático medio - MSE: 0.25984251968503935  
Coeficiente de determinación - R2: 0.19539236861051112

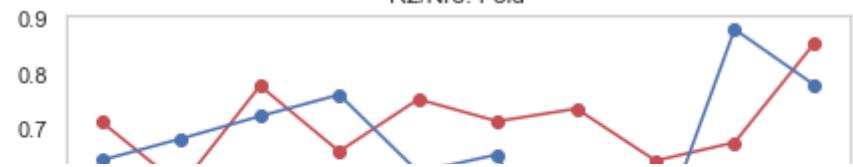
CART\_Reg:  
Error absoluto medio - MAE: 0.06978815148106486  
Error cuadrático medio - MSE: 0.09489327226953773  
Coeficiente de determinación - R2: 0.7061610581357574

RFR:  
Error absoluto medio - MAE: 0.0697244094488189  
Error cuadrático medio - MSE: 0.05579133858267717  
Coeficiente de determinación - R2: 0.8272409887209022





RFR  
R2/Nro. Fold



```
In [149]: def models_test(models=[], train_x=None, train_y=None, cv_sample=cv_sel, jobs=2, model_scoring="r2", model_type=np.nan):  
  
    for name, model in models:  
        # cross_val_score Kfold  
        cv_results = cross_val_score(model,  
                                      train_x,  
                                      train_y,  
                                      cv=cv_sel,  
                                      n_jobs=jobs,  
                                      scoring=model_scoring  
        )  
  
        # cross_val_score results  
        results.append(cv_results)  
        names.append(name)  
  
        sc_resul = cv_results.mean()  
        std_resul = cv_results.std()  
  
        md_name = str(model)[0:str(model).find("(")]  
  
        # Se agregan resultados para el reporte  
        model_resul['ModelName'].append(md_name)  
        model_resul['ModelCode'].append(name)  
        model_resul['ModelType'].append(model_type)  
        model_resul['Scoring'].append(model_scoring)  
        model_resul['ModelScore'].append(round(sc_resul, 3))  
        model_resul['ModelSTD'].append(round(std_resul, 3))  
  
        print('%s: Puntuaciones: %f - Desviación típica:(%f)' %(name, sc_resul, std_resul))  
  
def graphResults(results=results, labels=names):  
    print("\n")  
    plt.clf()  
    # Grafica - Boxplot  
    fig3, ax3 = plt.subplots(figsize=(12, 6))  
    ax3.margins(0.05)  
    ax3.set_title(  
        f'Evaluación de los resultado de los diferentes métodos predictivos')  
    green_diamond = dict(markerfacecolor='b', marker='o')  
    ax3.boxplot(results, labels=names, flierprops=green_diamond,
```

```
        autorange=True, vert=False)
plt.show()
print("\n\n")
```

In [150]:

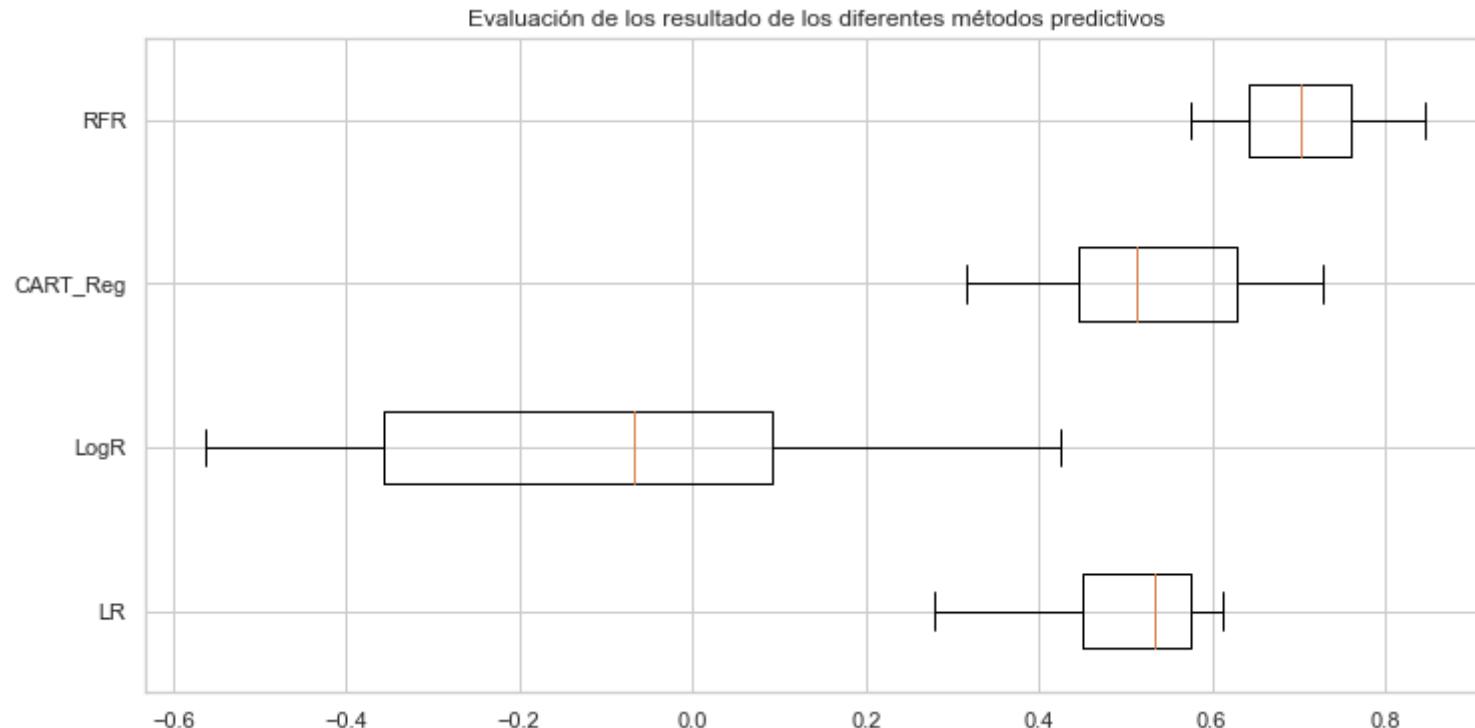
```
results = []
names= []

# Resultados cross_val_score / scoring="r2"
models_test(models_reg, train_x, train_y, cv_sample = cv_sel, jobs=2, model_scoring="r2", model_type="Regression")

# Graficar Resultados
graphResults(results=results, labels=names)
```

LR: Puntuaciones: 0.500528 - Desviación típica:(0.097017)  
LogR: Puntuaciones: -0.079889 - Desviación típica:(0.320451)  
CART\_Reg: Puntuaciones: 0.536911 - Desviación típica:(0.126592)  
RFR: Puntuaciones: 0.701966 - Desviación típica:(0.077785)

<Figure size 1080x360 with 0 Axes>



## **8.7 Modelos de Clasificación**

```
In [151]: # Resultados cross_val_score / scoring="accuracy"
results = []
names= []
models_test(models_clf, train_x, train_y, cv_sample = cv_sel, jobs=2, model_scoring="accuracy", model_type="Classifier")

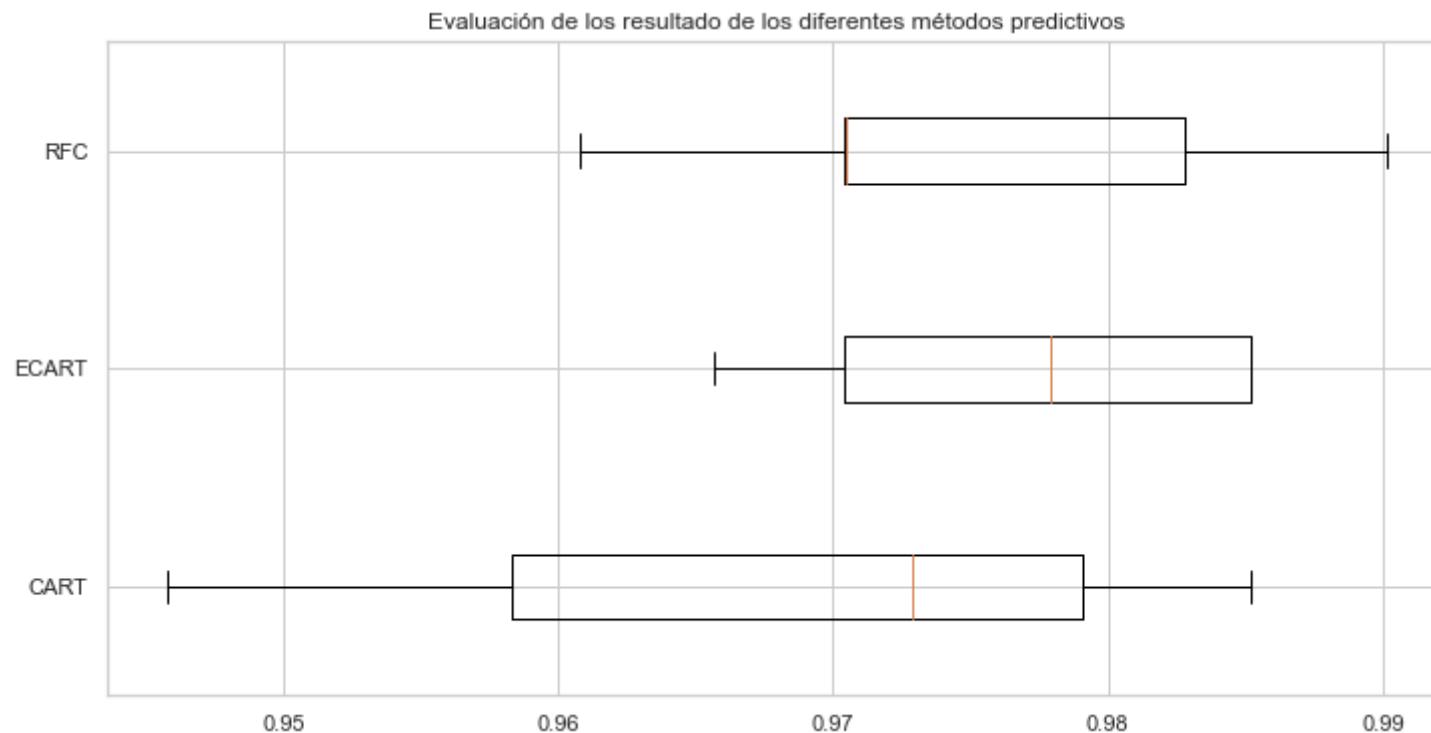
# Graficar Resultados
graphResults(results=results, labels=names)
```

CART: Puntuaciones: 0.969502 - Desviación típica:(0.012761)

ECART: Puntuaciones: 0.977366 - Desviación típica:(0.007350)

RFC: Puntuaciones: 0.974911 - Desviación típica:(0.008620)

<Figure size 1080x360 with 0 Axes>



In [152]:

```
predicted_df=pd.DataFrame()
predicted_df['Tiempo']= df['Fecha']
predicted_df['anomalia'] = df['anomalia']
# predicted_df['Prediccion']= clf_arboles.predict(X)
# predicted_df['Prediccion'] = models_reg[1][1].predict(X)
clfm = models_clf[2][1].fit(train_x, train_y)
clfm_pred = clfm.predict(X)
predicted_df['Prediccion'] = clfm_pred

# Se convierte la salida a (-1 , 1)
if len(pd.unique(predicted_df['Prediccion'])) > 2:
    predicted_df['Prediccion'] = np.where(predicted_df['Prediccion'] > 0.55, 1, -1)

# predicted_df.reset_index(inplace=True)
predicted_df.sample(15)
```

Out[152]:

|      | Tiempo              | anomalia | Prediccion |
|------|---------------------|----------|------------|
| 875  | 2021-03-15 15:50:05 | 1        | 1          |
| 1680 | 2021-03-18 23:10:10 | 1        | 1          |
| 1075 | 2021-03-16 11:00:06 | 1        | 1          |
| 551  | 2021-03-14 07:50:03 | 1        | 1          |
| 783  | 2021-03-15 06:50:05 | 1        | 1          |
| 439  | 2021-03-13 20:00:03 | 1        | 1          |
| 1841 | 2021-03-19 14:00:11 | 1        | 1          |
| 917  | 2021-03-15 19:30:05 | 1        | 1          |
| 1090 | 2021-03-16 12:20:06 | 1        | 1          |
| 2102 | 2021-03-20 14:55:12 | 1        | 1          |
| 2144 | 2021-03-20 19:00:13 | 1        | 1          |
| 2158 | 2021-03-20 20:20:13 | 1        | 1          |
| 2506 | 2021-03-22 06:45:15 | 1        | 1          |
| 1919 | 2021-03-19 21:45:11 | 1        | 1          |
| 227  | 2021-03-12 23:35:01 | 1        | 1          |

## 8.8 Reporte de resultados obtenidos

```
In [153]: mdl_resul=pd.DataFrame(model_resul)  
mdl_resul
```

Out[153]:

|   | ModelName              | ModelCode | ModelType  | Scoring  | ModelScore | ModelSTD |
|---|------------------------|-----------|------------|----------|------------|----------|
| 0 | LinearRegression       | LR        | Regression | r2       | 0.501      | 0.097    |
| 1 | LogisticRegression     | LogR      | Regression | r2       | -0.080     | 0.320    |
| 2 | DecisionTreeRegressor  | CART_Reg  | Regression | r2       | 0.537      | 0.127    |
| 3 | RandomForestRegressor  | RFR       | Regression | r2       | 0.702      | 0.078    |
| 4 | DecisionTreeClassifier | CART      | Classifier | accuracy | 0.970      | 0.013    |
| 5 | ExtraTreesClassifier   | ECART     | Classifier | accuracy | 0.977      | 0.007    |
| 6 | RandomForestClassifier | RFC       | Classifier | accuracy | 0.975      | 0.009    |

## 9 MÉTRICAS DE EVALUACIÓN MODELOS DE CLASIFICACIÓN

### 9.1 Medidas de la precisión de la clasificación

1. Accuracy
2. Precision
3. Recall
4. Specificity

Verdadero positivo: Interpretación: predijiste positivo y es cierto.

Verdadero negativo: Interpretación: predijiste negativo y es cierto.

Falso positivo: (error tipo 1) Interpretación: predijiste positivo y es falso.

Falso negativo: (Error de tipo 2) Interpretación: predijiste negativo y es falso.

**Abreviaturas:**

- **VP** = Verdaderos positivos | **TP** = True Positive

- **VN** = Verdaderos negativos | **TN** = True Negative
- **FP** = Falsos positivos | **FP** = False Positive
- **FN** = Falsos negativos. | **FN** = False Negative

### 9.1.1 Matriz de confusión

Es útil para medir la Sensibilidad, la precisión, la especificidad, la exactitud y las curvas AUC-ROC.

### 9.1.2 Exactitud (Accuracy) - ACC

Es el numero total de predicciones correctas dividido entre el numero total de predicciones, que es el porcentaje de la data clasificada correctamente. La exactitud es la fracción de predicciones que el modelo realizó correctamente.

|        |   | Prediction             |                        |
|--------|---|------------------------|------------------------|
|        |   | +                      | -                      |
| Actual | + | TP<br>(True Positive)  | FN<br>(False Negative) |
|        | - | FP<br>(False Positive) | TN<br>(True Negative)  |

$$\text{Exactitud} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{VP + VN}{TotalP + TotalN}$$

Ej.

$$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 1 \\ 1 & 2 & 12 \end{bmatrix}$$

$$\text{Exactitud} = \frac{5 + 3 + 12}{24} = 0.833$$

### Tasa de Error Rate Razón de falsos positivos (FPR) = 1 - SPC

En general, que porcentaje de la data clasifica **incorrectamente**. La tasa de error es porcentaje de instancias del conjunto de datos de pruebas que son clasificadas **incorrectamente**

$$\text{Tasa de Error Rate} = (1 - \text{Accuracy}) = \frac{FP + FN}{TotalP + TotalN}$$

### 9.1.3 Precision

Cuando predicen positivos, que porcentaje clasifica correctamente. Cuántas muestras clasificadas como clase positiva (TP + FP) pertenecen realmente a la clase positiva (TP).

|        |   | Precision              |                        |
|--------|---|------------------------|------------------------|
|        |   | Prediction             |                        |
|        |   | +                      | -                      |
| Actual | + | TP<br>(True Positive)  | FN<br>(False Negative) |
|        | - | FP<br>(False Positive) | TN<br>(True Negative)  |

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{TP}{\text{Total clasificados positivos}}$$

**Idealmente**, queremos que la **precisión** sea lo más alta posible → **100%**

Ej.

$$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 1 \\ 1 & 2 & 12 \end{bmatrix}$$

$$\text{Precision}_{\text{clase}_1} = \frac{5}{5 + 0 + 1} = 0.8333$$

$$\text{Precision}_{\text{clase}_2} = \frac{3}{0 + 3 + 2} = 0.6$$

$$\text{Precision}_{\text{clase}_3} = \frac{12}{0 + 1 + 12} = 0.92$$

#### 9.1.4 Sensibilidad - True Positive Rate - Razón de Verdaderos Positivos (TPR)

**Synonyms:** Recall; Sensitivity

La tasa de verdaderos positivos también se conoce como sensibilidad y se calcula como el número de verdaderos positivos dividido por la suma del número de verdaderos positivos y el número de falsos negativos. Describe qué tan bueno es el modelo para predecir la clase positiva cuando el resultado real es positivo.

En otras palabras, mide la capacidad del clasificador para detectar la clase en una población con instancias de esa clase, la **proporción de casos positivos bien detectados**.

|        |   | Prediction             |                        |
|--------|---|------------------------|------------------------|
|        |   | +                      | -                      |
| Actual | + | TP<br>(True Positive)  | FN<br>(False Negative) |
|        | - | FP<br>(False Positive) | TN<br>(True Negative)  |

$$\text{Recall}(TPR) = \frac{TP}{TP + FN} = \frac{TP}{TotalP}$$

Ej.

$$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 1 \\ 1 & 2 & 12 \end{bmatrix}$$

$$\text{Sensibilidad}(TPR)_{clase_1} = \frac{5}{5 + 0 + 0} = 1$$

$$\text{Sensibilidad}(TPR)_{clase_2} = \frac{3}{0 + 3 + 1} = 0.75$$

$$\text{Sensibilidad}(TPR)_{clase_3} = \frac{12}{1 + 2 + 12} = 0.85$$

### 9.1.5 Especificidad (specificity)- True Negative Rate - Razón de Verdaderos Negativos (TNR)

La tasa de falsos positivos se calcula como el número de falsos positivos dividido por la suma del número de falsos positivos y el número de verdaderos negativos. También se denomina tasa de falsas alarmas, ya que resume la frecuencia con la que se predice una clase positiva cuando el resultado real es negativo.

Mide la capacidad del clasificador para descartar instancias que no pertenecen a la clase en una población con instancias de esa clase, la proporción de casos negativos correctamente detectados.

$$\text{False Positive Rate} = 1 - Specificity$$

$$\text{Specificity}(TNR) = \frac{TN}{TN + FP} = \frac{TN}{TotalN}$$

$$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 1 \\ 1 & 2 & 12 \end{bmatrix}$$

Eje

**Clase\_1**

$$\text{Specificity}(TNR)_{clase_1} = \frac{3 + 1 + 2 + 12}{3 + 1 + 2 + 12 + 1} = 0.94$$

## **Clase\_2**

$$\begin{bmatrix} TN(5) & FP(0) & TN(0) \\ FN(0) & TP(3) & FN(1) \\ TN(1) & FP(2) & TN(12) \end{bmatrix}$$

$$\text{Specificity(TNR)}_{clase_2} = \frac{5 + 0 + 1 + 12}{5 + 0 + 1 + 12 + 2} = 0.9$$

$$\text{Specificity(TNR)}_{clase_3} = \frac{5 + 3}{5 + 3 + 1} = 0.88$$

*Es lo contrario de lo que es Recall.*

### **9.1.6 Ratio de falsos positivos (FPR – False Positive Rate)**

Proporción de casos negativos que el clasificador detecta como positivos.

$$FPR = \frac{FP}{FP + TN}$$

Ej.

$$\text{FP Rate}_{clase_1} = \frac{0 + 1}{0 + 1 + 3 + 1 + 2 + 12} = 0.053$$

## **Clase\_2**

$$\begin{bmatrix} TN(5) & FP(0) & TN(0) \\ FN(0) & TP(3) & FN(1) \\ TN(1) & FP(2) & TN(12) \end{bmatrix}$$

$$\text{FP Rate}_{clase_2} = \frac{0 + 2}{0 + 2 + 5 + 0 + 1 + 12} = 0.1$$

$$\text{FP Rate}_{clase_3} = \frac{0 + 1}{0 + 1 + 5 + 0 + 0 + 3} = 0.111$$

### **9.1.7 F1 Score**

La puntuación F1, también conocida como puntuación F equilibrada o medida F, es la medida armónica de precisión y sensibilidad, se puede interpretar como un promedio ponderado de la precisión y la recuperación, donde una puntuación F1 alcanza su mejor valor en 1 y la peor puntuación en 0. La contribución relativa de precisión y recuperación a la puntuación F1 son iguales.

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

La puntuación F1 se puede interpretar como un promedio ponderado de la precisión y la recuperación, donde una puntuación F1 alcanza su mejor valor en 1 y la peor puntuación en 0.

En el caso de múltiples clases y múltiples etiquetas, este es el promedio de la puntuación F1 de cada clase con una ponderación que depende del parámetro promedio.

### 9.1.8 Scores

**accuracy\_score** precision\_score devuelve el **porcentaje de etiquetas que predijo correctamente**. Por lo cual, si nuestro conjunto de datos tienen 1000 etiquetas y predijo 980 con precisión, quiere decir que obtiene una puntuación del 98%.

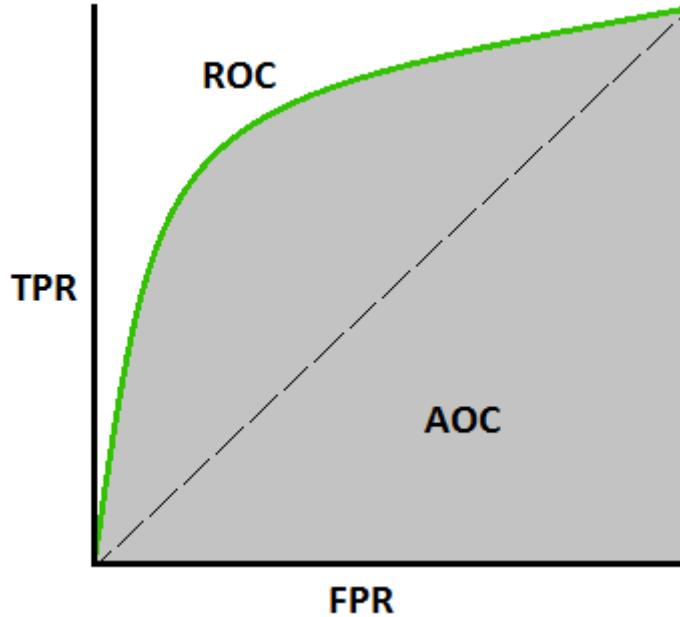
**balance\_accuracy\_score** balance\_accuracy\_score devuelve la **precisión promedio por clase**. Si tenemos 1000 etiquetas son de 2 clases y hay 750 observaciones en la clase 1 y 250 en la clase 2. Si el modelo no predice 10 en cada clase, tiene una precisión de  $740/750 = 98.7\%$  en la clase 1 y  $240/250 = 96\%$  en la clase 2. Balance\_accuracy\_score volvería entonces  $(98.7\% + 96\%) / 2 = 97.35\%$ .

### 9.1.9 Curvas ROC - Curvas de rendimiento diagnóstico

La curva AUC - ROC es una medida de rendimiento para los problemas de clasificación en varios valores de umbral. ROC es una curva de probabilidad y AUC representa el grado o medida de separabilidad. Indica cuánto es capaz el modelo de distinguir entre clases.

La curva ROC se traza con TPR contra el FPR donde TPR está en el eje y y FPR está en el eje x.

Esta área posee un valor comprendido entre 0,5 y 1, donde 1 representa un valor diagnóstico perfecto y 0,5 es una prueba sin capacidad discriminatoria diagnóstica.



- TPR (True Positive Rate) / Recall /Sensitivity
- FPR (1 - Specificity)

A modo de guía para interpretar las curvas ROC se han establecido los siguientes intervalos para los valores de AUC:

| Rango       | Resultado                 |
|-------------|---------------------------|
| [0.5]       | Es como lanzar una moneda |
| [0.5, 0.6)  | Test malo                 |
| [0.6, 0.75) | Test regular              |
| [0.75, 0.9) | Test bueno                |
| [0.9, 0.97) | Test muy bueno            |
| [0.97, 1)   | Test excelente            |

Un clasificador sin habilidad es uno que no puede discriminar entre las clases y predeciría una clase aleatoria o una clase constante en todos los casos. Un modelo sin habilidad se representa en el punto (0.5, 0.5). Un modelo sin habilidad en cada umbral está representado por una línea diagonal desde la parte inferior izquierda del gráfico hasta la parte superior derecha y tiene un AUC de 0.5.

Un modelo con habilidad perfecta se representa en un punto (0,1). Un modelo con habilidad perfecta está representado por una línea que viaja desde la parte inferior izquierda de la trama hasta la parte superior izquierda y luego a través de la parte superior derecha.

Un operador puede trazar la curva ROC para el modelo final y elegir un umbral que proporcione un equilibrio deseable entre los falsos positivos y los falsos negativos.

## Referencias:

- [AUC \(<https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/>\)](https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/)
- [Clf evaluation metrics \(<https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226>\)](https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226)
- [Accuracy \(<https://developers.google.com/machine-learning/crash-course/classification/accuracy>\)](https://developers.google.com/machine-learning/crash-course/classification/accuracy)
- [Performance Metrics ML \(<https://medium.com/machine-learning-through-visuals/performance-metrics-machine-learning-through-visuals-84ebd7cdcc20>\)](https://medium.com/machine-learning-through-visuals/performance-metrics-machine-learning-through-visuals-84ebd7cdcc20)
- [metrics.f1\\_score \(\[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1\\\_score.html\]\(https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1\_score.html\)\)](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html)
- [balanced\\_accuracy\\_score \(\[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced\\\_accuracy\\\_score.html\]\(https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced\_accuracy\_score.html\)\)](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html)
- [Diff bw accuracy-score-and-accuracy-score \(<https://stackoverflow.com/questions/55548675/difference-between-balanced-accuracy-score-and-accuracy-score>\)](https://stackoverflow.com/questions/55548675/difference-between-balanced-accuracy-score-and-accuracy-score)
- [Evaluación de errores \(<https://aprendeia.com/evaluando-el-error-en-los-modelos-de-clasificacion-machine-learning/>\)](https://aprendeia.com/evaluando-el-error-en-los-modelos-de-clasificacion-machine-learning/)
- [scoring-parameter \(\[https://scikit-learn.org/stable/modules/model\\\_evaluation.html#scoring-parameter\]\(https://scikit-learn.org/stable/modules/model\_evaluation.html#scoring-parameter\)\)](https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter)
- [Curvas ROC \(\[https://es.wikipedia.org/wiki/Curva\\\_ROC\]\(https://es.wikipedia.org/wiki/Curva\_ROC\)\)](https://es.wikipedia.org/wiki/Curva_ROC)
- [Curvas ROC 2 \(<https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>\)](https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62)
- [Confusion Matrix \(<https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>\)](https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62)
- [imbalanced-classification \(<https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/>\)](https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/)

## 10 Reporte

In [154]:

```
pred_models = []
for name, model in models_clf:

    reg_clf_model = model
    reg_clf_model.fit(train_x, train_y)
    pred_model = reg_clf_model.predict(test_x)

    # Predicciones
    pred_models.append((name, pred_model))
```

In [155]:

```
from sklearn.metrics import auc as auc_
from sklearn.metrics import roc_curve

# Se agregan resultados para el reporte
mc_metrics = {'ModelName': [],  
               'AccuracyScore': [],  
               'BalancedAccuracyScore': [],  
               'f1_Score': [],  
               'ROC_AUC_Score': []  
             }  
  
for md in pred_models:  
    #     m_name, m_pred = pred_models[1]  
    m_name, m_pred = md  
    # Se transforma en salida binaria a los resultados de las regresiones.  
    if len(pd.unique(m_pred)) > 2:  
        m_pred = np.where(m_pred > 0.55, 1, -1)  
        # display(m_pred)  
    print(m_name)  
  
    cf_matrix = confusion_matrix(test_y, m_pred, labels=[-1, 1])  
  
    mc_res = show_mc_mtr(test_y, m_pred)  
  
    mfg = pd.DataFrame(mc_res)  
    display(mfg)  
    #     display(mfg.T)  
  
    acc_sc = accuracy_score(test_y, m_pred)  
    print(f"Accuracy Score: {acc_sc}")  
  
    bacc_sc = balanced_accuracy_score(test_y, m_pred)  
    print(f"Balanced_accuracy_score: {bacc_sc}")  
  
    f1 = f1_score(test_y, m_pred)  
    print(f"f1_score: {f1}")  
  
    # calculate AUC  
    auc = roc_auc_score(test_y, m_pred)  
    print('AUC: %.3f' % auc)  
  
    print('\nClassification Report: ')
    print(classification_report(test_y, m_pred))
```

```
# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(test_y, m_pred)
plot_roc_curve(fpr, tpr)

# Se agregan resultados para el reporte
mc_metrics['ModelName'].append(m_name)
mc_metrics['AccuracyScore'].append(acc_sc)
mc_metrics['BalancedAccuracyScore'].append(bacc_sc)
mc_metrics['f1_Score'].append(f1)
mc_metrics['ROC_AUC_Score'].append(auc)

#
# calculate precision-recall curve
# lr_precision, lr_recall, _ = precision_recall_curve(test_y, m_pred)
# f1, lr_auc = f1_score(test_y, m_pred), auc_(lr_recall, lr_precision)

#
# summarize scores
# print('clf: f1=% .3f auc=% .3f' % (f1, lr_auc))

# Grafica - Boxplot
fig4, ax4 = plt.subplots(figsize=(12, 6))
ax4.margins(0.05)
ax4.set_title(f'Matriz de confusión predicción - {m_name}')
ax4 = sns.heatmap(cf_matrix, annot=True, fmt='0')
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()

#     metrics.plot_roc_curve(test_y, m_pred)
skplt.metrics.plot_confusion_matrix(test_y, m_pred, normalize=False)
plt.show()

# Grafica - Boxplot
fig5, ax5 = plt.subplots(figsize=(12, 6))
ax5.margins(0.05)
ax5.set_title(f'Matriz de confusión predicción - {m_name}')
ax5 = sns.heatmap(cf_matrix/np.sum(cf_matrix),
                  annot=True, fmt='%.2%', cmap='Blues')
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()

# https://www.statology.org/plot-roc-curve-python/
```

---

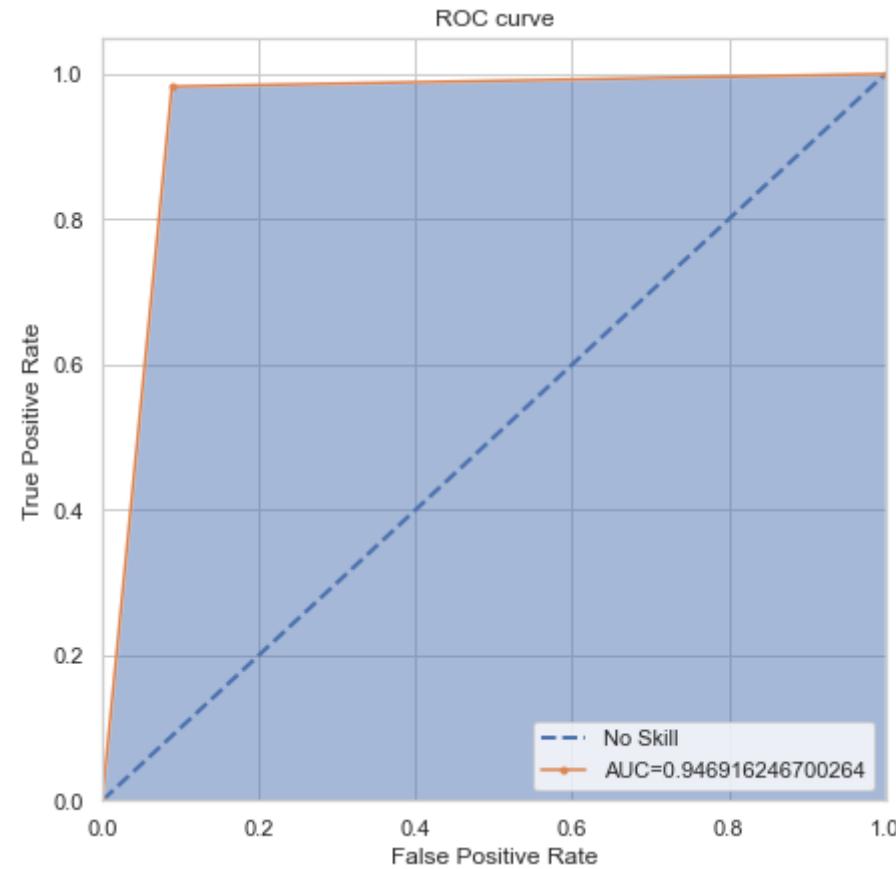
## CART

|                                | Clase_0 | Clase_1 |
|--------------------------------|---------|---------|
| <b>TP</b>                      | 41.000  | 455.000 |
| <b>FP</b>                      | 8.000   | 4.000   |
| <b>FN</b>                      | 4.000   | 8.000   |
| <b>TN</b>                      | 455.000 | 41.000  |
| <b>Recall</b>                  | 0.911   | 0.983   |
| <b>Specificity</b>             | 0.983   | 0.911   |
| <b>PositivePredictiveValue</b> | 0.837   | 0.991   |
| <b>NegativePredictiveValue</b> | 0.991   | 0.837   |
| <b>FalsePositiveRate</b>       | 0.017   | 0.089   |
| <b>FalseNegativeRate</b>       | 0.089   | 0.017   |
| <b>Accuracy</b>                | 0.976   | 0.976   |
| <b>ErrorRate</b>               | 0.024   | 0.024   |

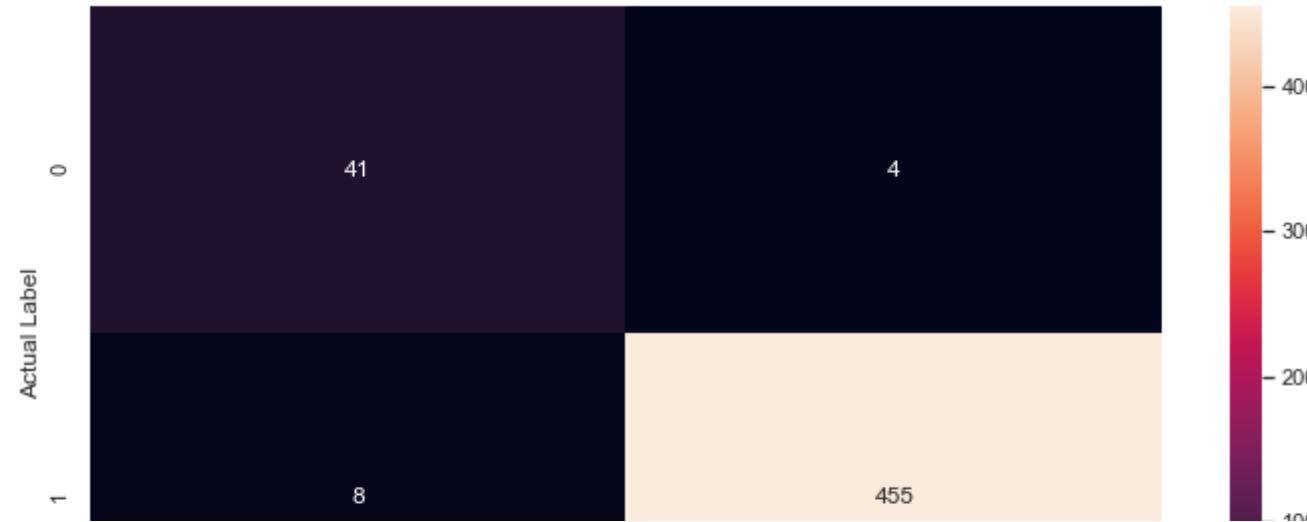
Accuracy Score: 0.9763779527559056  
Balanced\_accuracy\_score: 0.946916246700264  
f1\_score: 0.9869848156182213  
AUC: 0.947

### Classification Report:

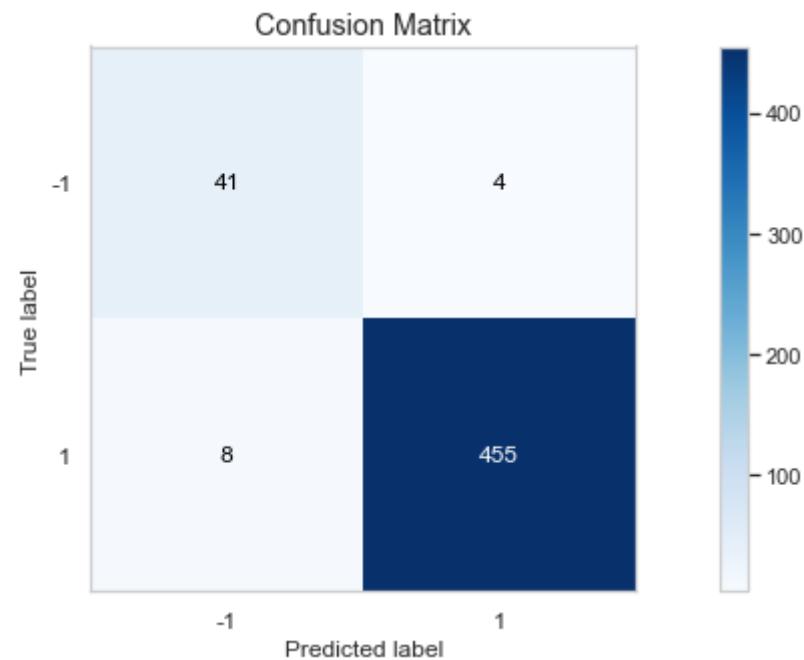
|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| -1           | 0.84      | 0.91   | 0.87     | 45      |
| 1            | 0.99      | 0.98   | 0.99     | 463     |
| accuracy     |           |        | 0.98     | 508     |
| macro avg    | 0.91      | 0.95   | 0.93     | 508     |
| weighted avg | 0.98      | 0.98   | 0.98     | 508     |

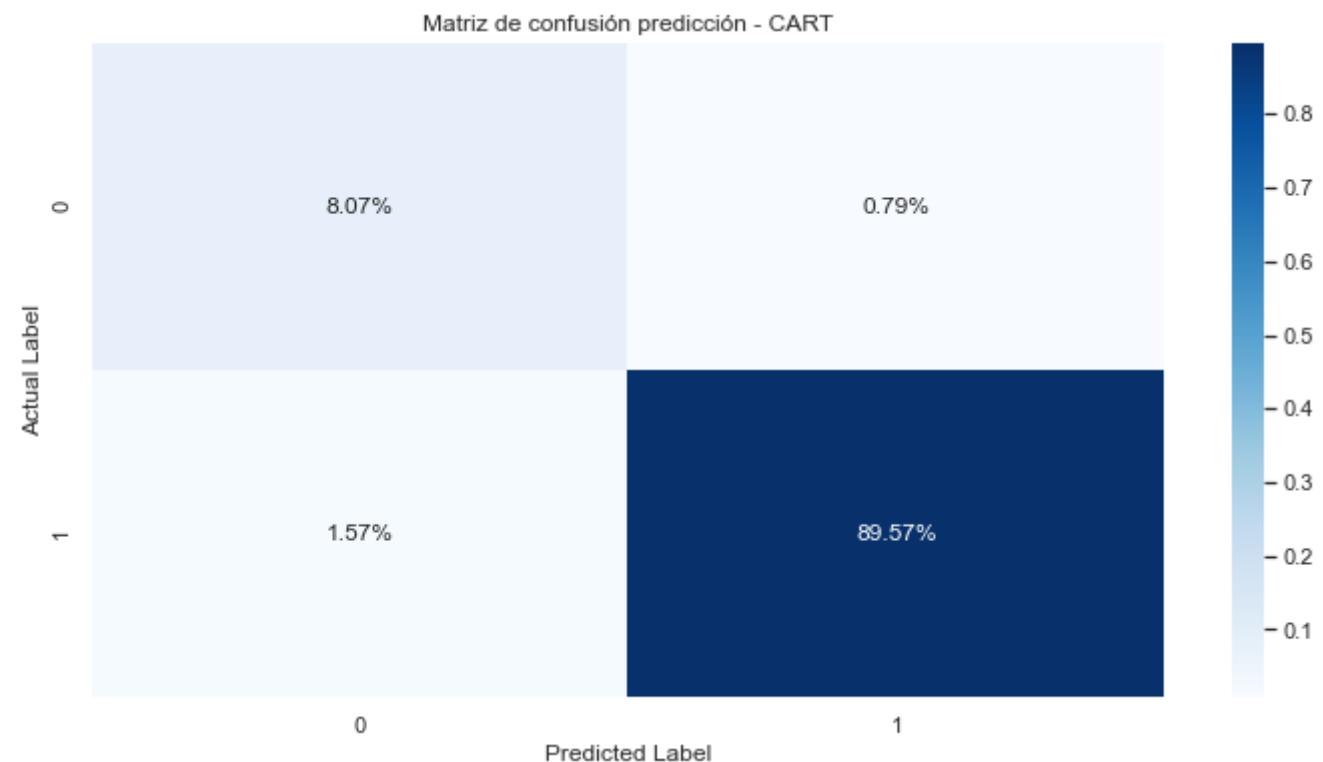


Matriz de confusión predicción - CART



Confusion Matrix





ECART

|                                | Clase_0 | Clase_1 |
|--------------------------------|---------|---------|
| <b>TP</b>                      | 37.000  | 462.000 |
| <b>FP</b>                      | 1.000   | 8.000   |
| <b>FN</b>                      | 8.000   | 1.000   |
| <b>TN</b>                      | 462.000 | 37.000  |
| <b>Recall</b>                  | 0.822   | 0.998   |
| <b>Specificity</b>             | 0.998   | 0.822   |
| <b>PositivePredictiveValue</b> | 0.974   | 0.983   |
| <b>NegativePredictiveValue</b> | 0.983   | 0.974   |

|                          | Clase_0 | Clase_1 |
|--------------------------|---------|---------|
| <b>FalsePositiveRate</b> | 0.002   | 0.178   |
| <b>FalseNegativeRate</b> | 0.178   | 0.002   |
| <b>Accuracy</b>          | 0.982   | 0.982   |

Accuracy Score: 0.9822834645669292

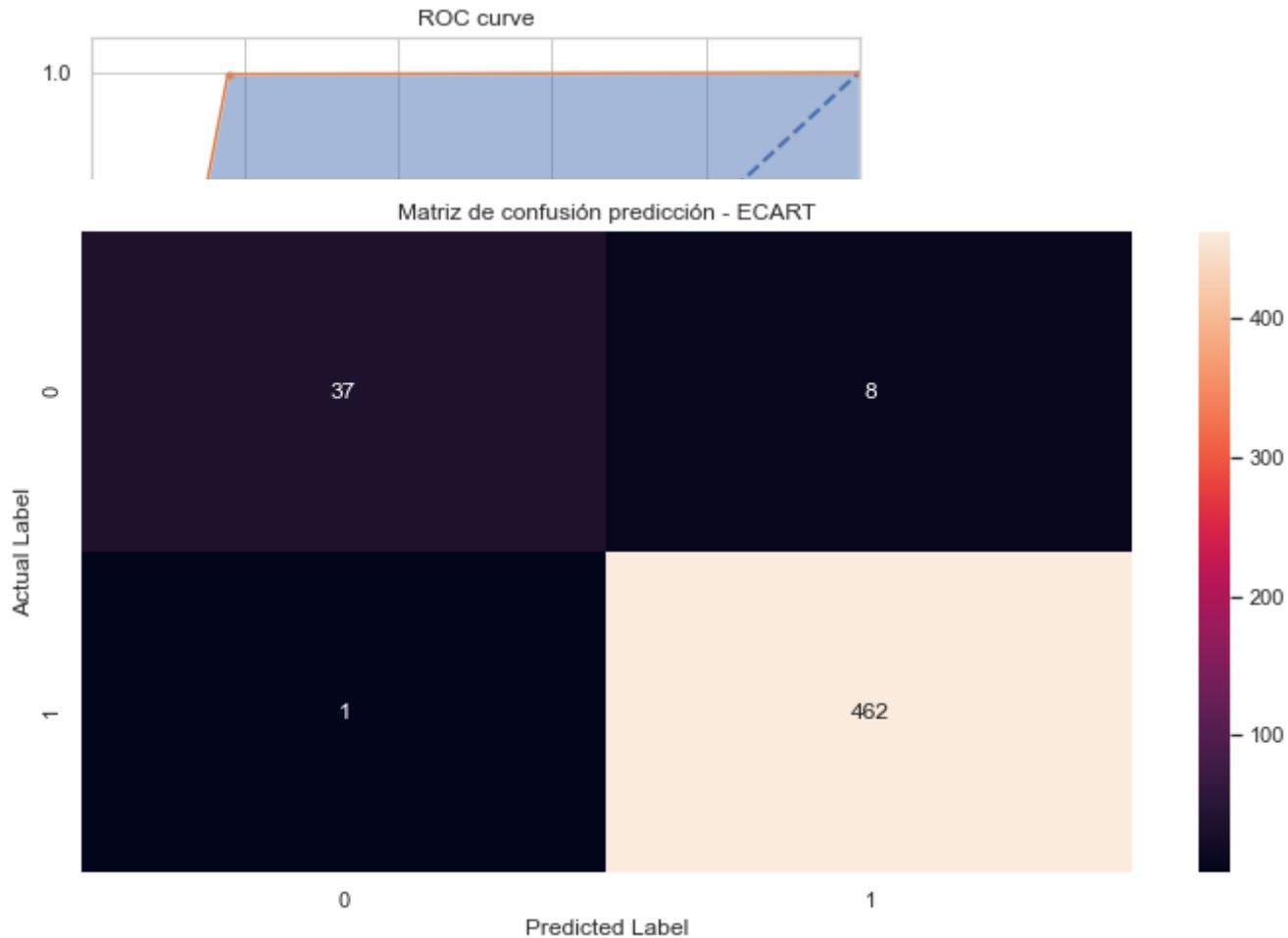
Balanced\_accuracy\_score: 0.9100311975041997

f1\_score: 0.9903536977491962

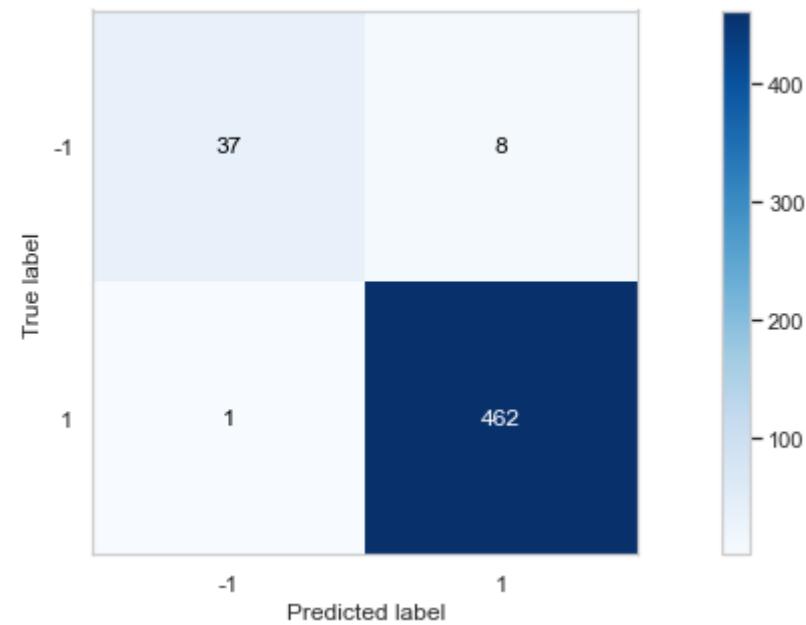
AUC: 0.910

Classification Report:

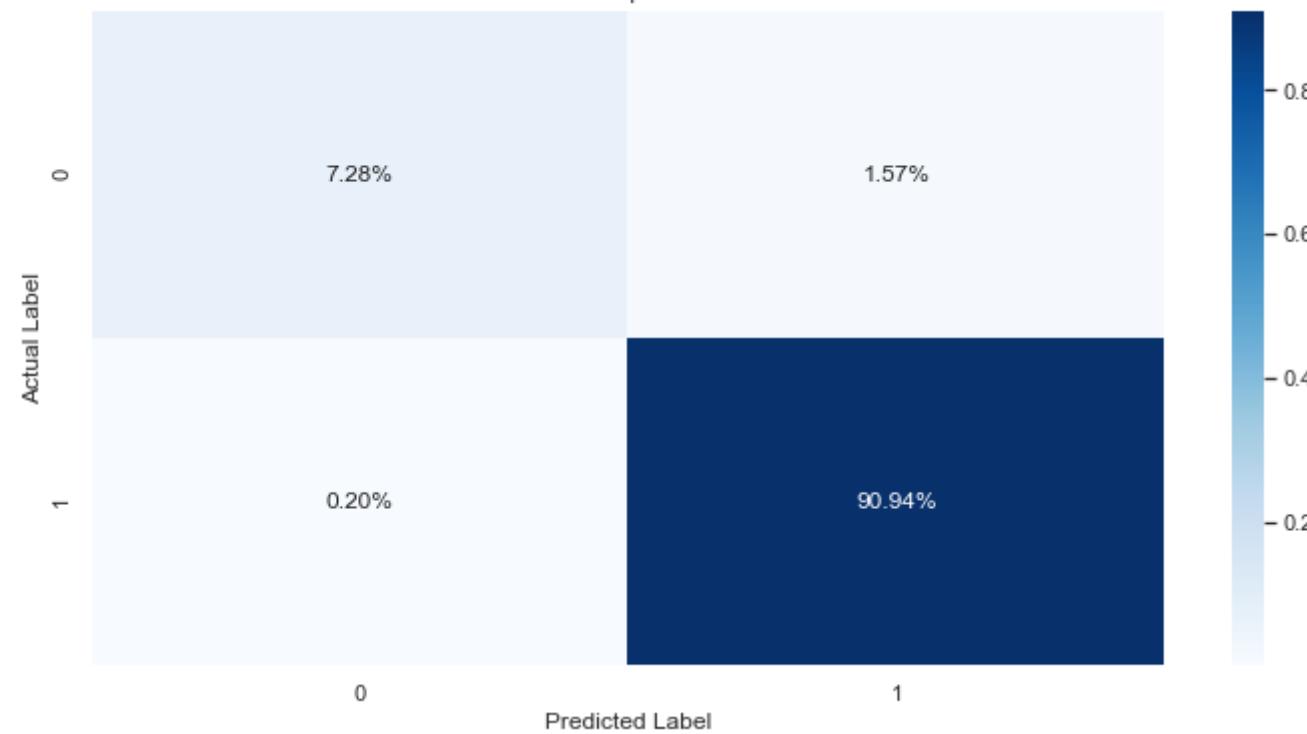
|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| -1           | 0.97      | 0.82   | 0.89     | 45      |
| 1            | 0.98      | 1.00   | 0.99     | 463     |
| accuracy     |           |        | 0.98     | 508     |
| macro avg    | 0.98      | 0.91   | 0.94     | 508     |
| weighted avg | 0.98      | 0.98   | 0.98     | 508     |



Confusion Matrix



Matriz de confusión predicción - ECART



RFC

|                                | Clase_0 | Clase_1 |
|--------------------------------|---------|---------|
| <b>TP</b>                      | 36.000  | 460.000 |
| <b>FP</b>                      | 3.000   | 9.000   |
| <b>FN</b>                      | 9.000   | 3.000   |
| <b>TN</b>                      | 460.000 | 36.000  |
| <b>Recall</b>                  | 0.800   | 0.994   |
| <b>Specificity</b>             | 0.994   | 0.800   |
| <b>PositivePredictiveValue</b> | 0.923   | 0.981   |
| <b>NegativePredictiveValue</b> | 0.981   | 0.923   |
| <b>FalsePositiveRate</b>       | 0.006   | 0.200   |
| <b>FalseNegativeRate</b>       | 0.200   | 0.006   |
| <b>Accuracy</b>                | 0.976   | 0.976   |
| <b>ErrorRate</b>               | 0.024   | 0.024   |

Accuracy Score: 0.9763779527559056

Balanced\_accuracy\_score: 0.8967602591792656

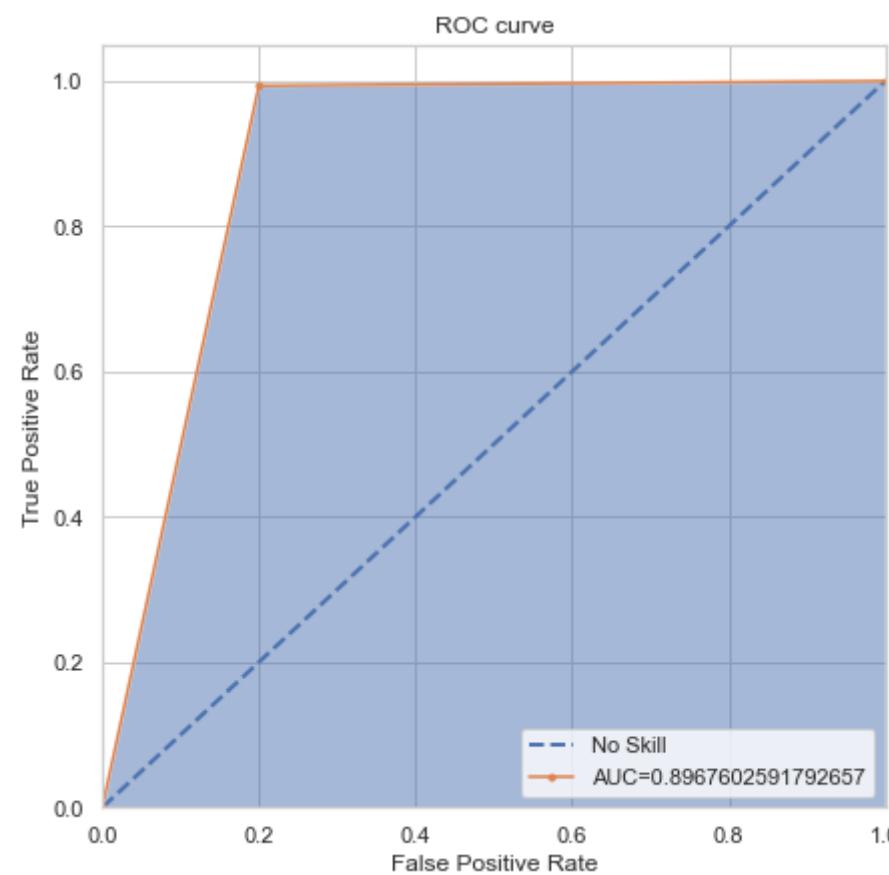
f1\_score: 0.9871244635193133

AUC: 0.897

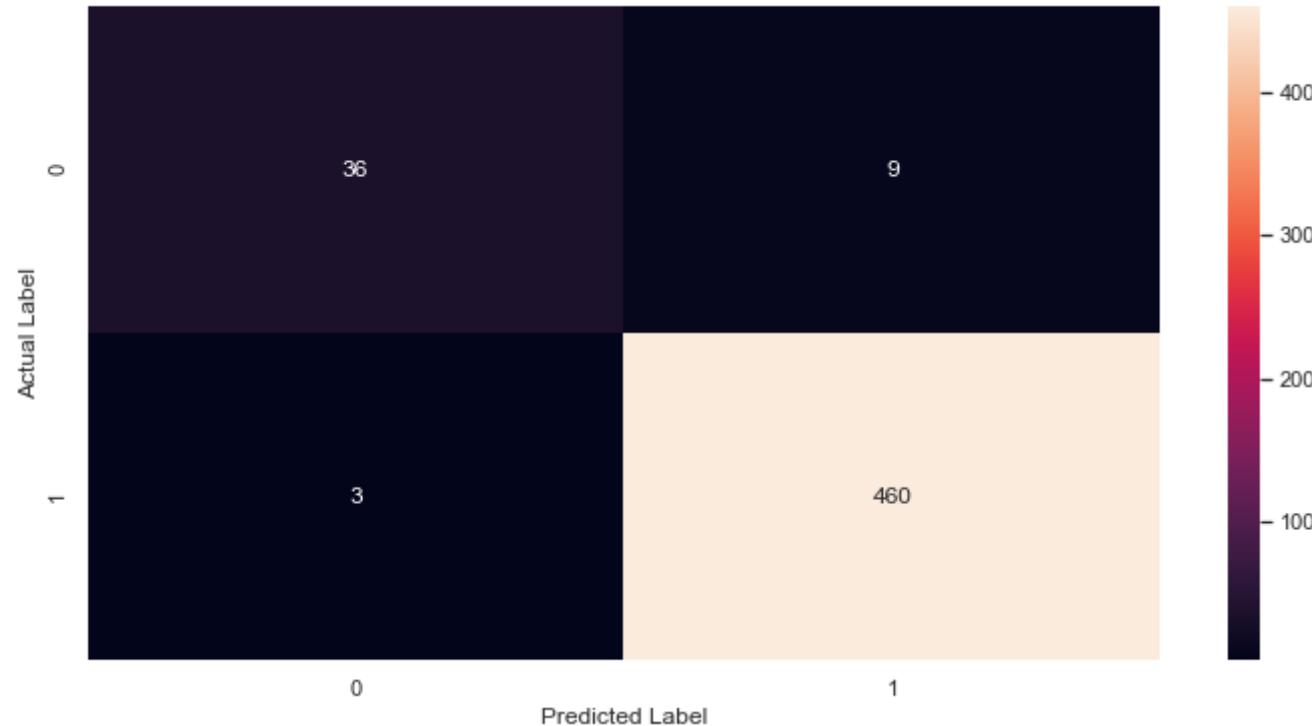
Classification Report:

|          | precision | recall | f1-score | support |
|----------|-----------|--------|----------|---------|
| -1       | 0.92      | 0.80   | 0.86     | 45      |
| 1        | 0.98      | 0.99   | 0.99     | 463     |
| accuracy |           | 0.98   |          | 508     |

|              |      |      |      |     |
|--------------|------|------|------|-----|
| macro avg    | 0.95 | 0.90 | 0.92 | 508 |
| weighted avg | 0.98 | 0.98 | 0.98 | 508 |



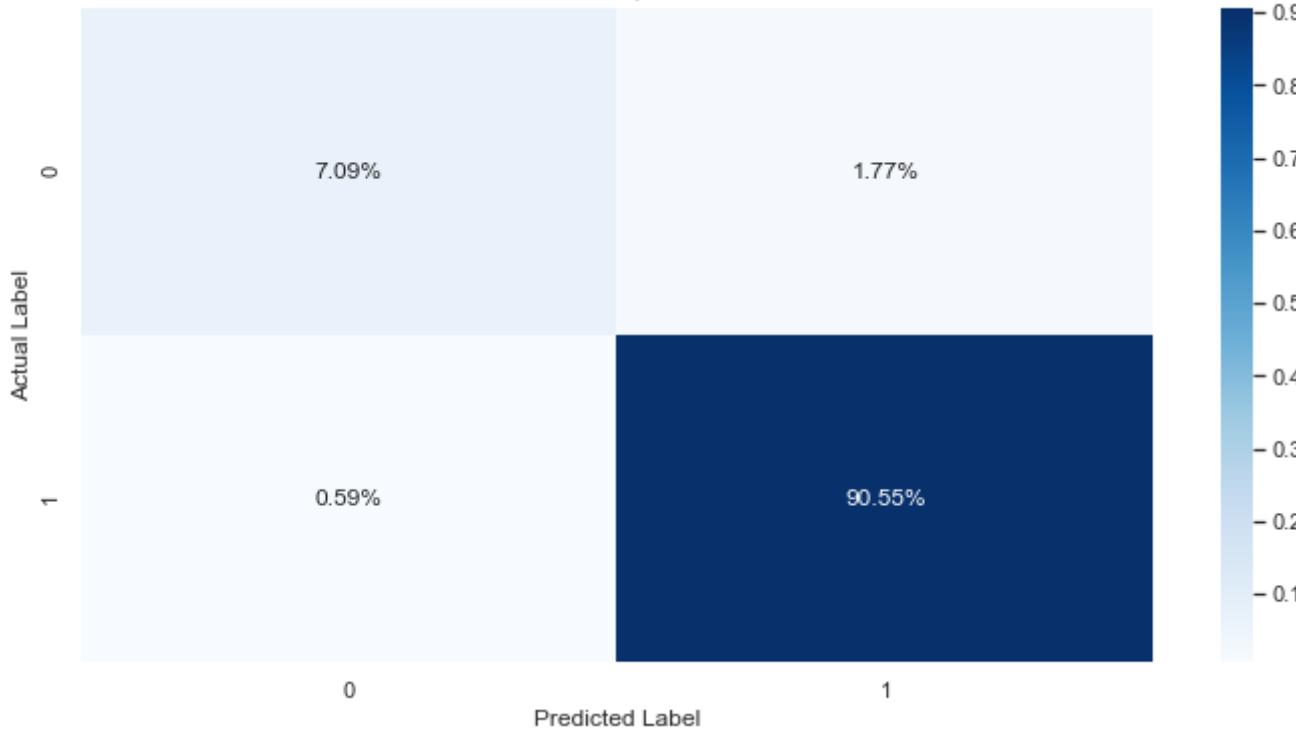
Matriz de confusión predicción - RFC



Confusion Matrix



Matriz de confusión predicción - RFC





```
In [158]: mgg = pd.DataFrame(mc_metrics)
display(mgg)
```

|   | ModelName | AccuracyScore | BalancedAccuracyScore | f1_Score | ROC_AUC_Score |
|---|-----------|---------------|-----------------------|----------|---------------|
| 0 | CART      | 0.976         | 0.947                 | 0.987    | 0.947         |
| 1 | ECART     | 0.982         | 0.910                 | 0.990    | 0.910         |
| 2 | RFC       | 0.976         | 0.897                 | 0.987    | 0.897         |

```
In [160]: # ref: https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226
# ref: https://medium.com/swlh/generating-a-dataset-with-gans-1e994ff633fd
# ref: https://github.com/kjschmidt913/GAN-generate-data/blob/master/Deep_Learning_From_Scratch_Original.ipynb
# ref: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/
# ref: https://towardsdatascience.com/synthetic-time-series-data-a-gan-approach-869a984f2239
```

## 10.1 Técnicas de remuestreo - Clase minoritaria de sobremuestreo

El sobremuestreo se puede definir como agregar más copias de la clase minoritaria.

```
In [161]: # Ref: https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18
# Ref: https://imbalanced-learn.org/stable/
# Ref: https://towardsdatascience.com/guide-to-classification-on-imbalanced-datasets-d6653aa5fa23
# Ref: https://www.datacamp.com/community/tutorials/random-forests-classifier-python
```

In [162]:

```
from sklearn.utils import resample
from sklearn.metrics import recall_score

# concatenate our training data back together
X1 = pd.concat([train_x, train_y], axis=1)

# separate minority and majority classes
reg_normal = X1[X1.anomalia==1]
reg_anomalo = X1[X1.anomalia==0]

# upsample minority
anomalia_upsampled = resample(reg_anomalo,
                               replace=True, # sample with replacement
                               n_samples=len(reg_normal), # match number in majority class
                               random_state=semilla_aleatoria) # reproducible results

# combine majority and upsampled minority
upsampled = pd.concat([reg_normal, anomalia_upsampled])

# check new class counts
display(upsampled.anomalia.value_counts())

y_train = upsampled.anomalia
x_train = upsampled.drop('anomalia', axis=1)

# Resultados cross_val_score / scoring="accuracy"
results = []
names = []
models_test(models_clf, x_train, y_train, cv_sample = cv_sel, jobs=2, model_scoring="accuracy", model_type="Classifier")

# Graficar Resultados
graphResults(results=results, labels=names)

upsampled_clf_model = models_clf[1][1]
display(upsampled_clf_model)
upsampled_clf_model.fit(x_train, y_train)
upsampled_predict = upsampled_clf_model.predict(test_x)

display("accuracy_score", accuracy_score(test_y, upsampled_predict))
```

```
display("f1_score", f1_score(test_y, upsampled_predict))

display("recall_score", recall_score(test_y, upsampled_predict))

upsampled_res = show_mc_mtr(test_y, upsampled_predict)

mmfg = pd.DataFrame(upsampled_res)
display(mmgf)

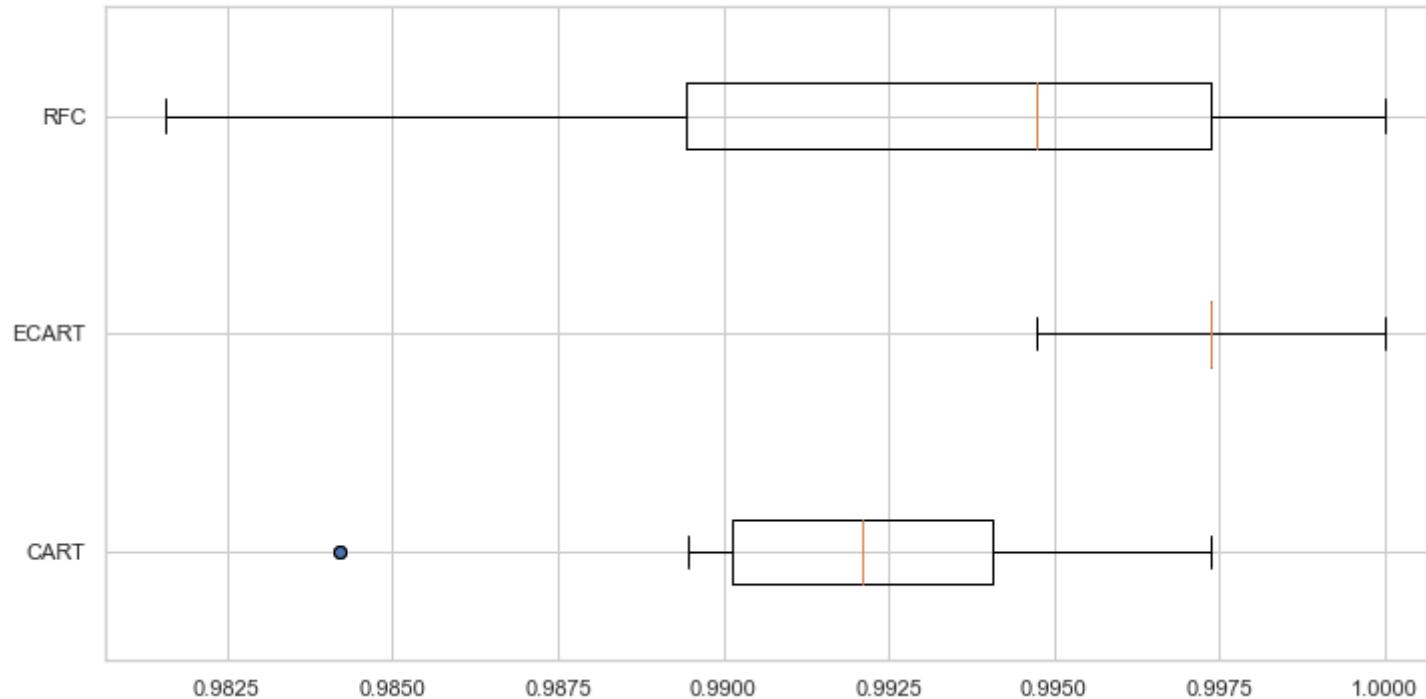
skplt.metrics.plot_confusion_matrix(test_y, upsampled_predict, normalize=False)
plt.show()
```

```
1    1899
-1    1899
Name: anomalia, dtype: int64
```

```
CART: Puntuaciones: 0.991312 - Desviación típica:(0.004084)
ECART: Puntuaciones: 0.997367 - Desviación típica:(0.001667)
RFC: Puntuaciones: 0.992891 - Desviación típica:(0.005527)
```

```
<Figure size 1080x360 with 0 Axes>
```

Evaluación de los resultado de los diferentes métodos predictivos



```
ExtraTreesClassifier(n_estimators=10, random_state=2021)
```

```
'accuracy_score'
```

```
0.9783464566929134
```

```
'f1_score'
```

```
0.9882352941176471
```

```
'recall_score'
```

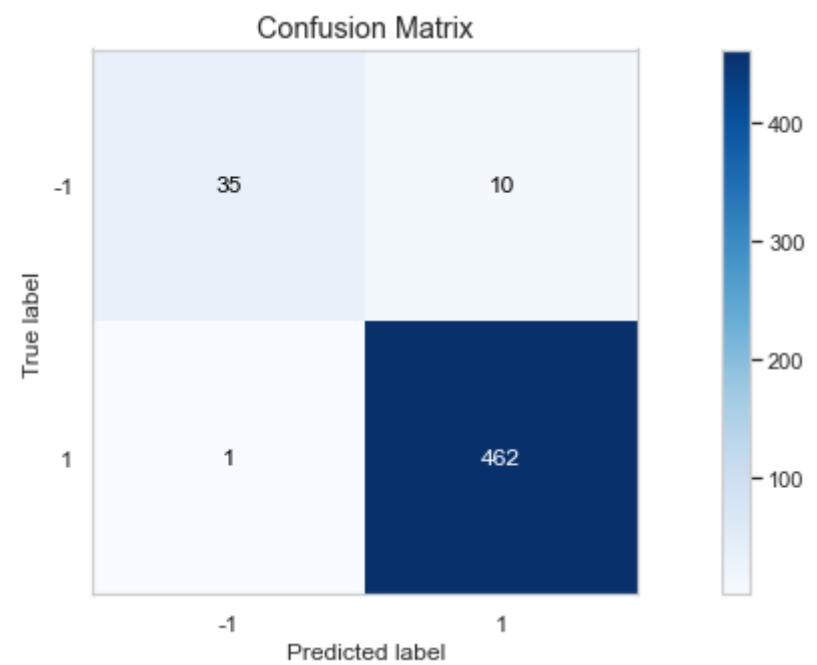
```
0.9978401727861771
```

|  | Clase_0 | Clase_1 |
|--|---------|---------|
|--|---------|---------|

|    |        |         |
|----|--------|---------|
| TP | 35.000 | 462.000 |
|----|--------|---------|

|    |       |        |
|----|-------|--------|
| FP | 1.000 | 10.000 |
|----|-------|--------|

|                                | Clase_0 | Clase_1 |
|--------------------------------|---------|---------|
| <b>FN</b>                      | 10.000  | 1.000   |
| <b>TN</b>                      | 462.000 | 35.000  |
| <b>Recall</b>                  | 0.778   | 0.998   |
| <b>Specificity</b>             | 0.998   | 0.778   |
| <b>PositivePredictiveValue</b> | 0.972   | 0.979   |
| <b>NegativePredictiveValue</b> | 0.979   | 0.972   |
| <b>FalsePositiveRate</b>       | 0.002   | 0.222   |
| <b>FalseNegativeRate</b>       | 0.222   | 0.002   |
| <b>Accuracy</b>                | 0.978   | 0.978   |
| <b>ErrorRate</b>               | 0.022   | 0.022   |



In [163]:

```
# upsample minority
anomalia_downsampled = resample(reg_normal,
                                replace=True, # sample with replacement
                                n_samples=len(reg_anomalo), # match number in majority class
                                random_state=semilla_aleatoria) # reproducible results

# combine
downsampled = pd.concat([reg_anomalo, anomalia_downsampled])

# check new class counts
display(downsampled.anomalia.value_counts())

y_train = downsampled.anomalia
x_train = downsampled.drop('anomalia', axis=1)

# Resultados cross_val_score / scoring="accuracy"
results = []
names= []
models_test(models_clf, x_train, y_train, cv_sample = cv_sel, jobs=2, model_scoring="accuracy", model_type="Classifier")

# Graficar Resultados
graphResults(results=results, labels=names)

display(upsampled_clf_model)
upsampled_clf_model.fit(x_train, y_train)
upsampled_predict = upsampled_clf_model.predict(test_x)

display("accuracy_score", accuracy_score(test_y, upsampled_predict))

display("f1_score", f1_score(test_y, upsampled_predict))

display("recall_score", recall_score(test_y, upsampled_predict))

upsampled_res = show_mc_mtr(test_y, upsampled_predict)

mmfg = pd.DataFrame(upsampled_res)
display(mmfg)

skplt.metrics.plot_confusion_matrix(test_y, upsampled_predict, normalize=False)
```

```
| plt.show()
```

```
| 1    133
```

```
| -1   133
```

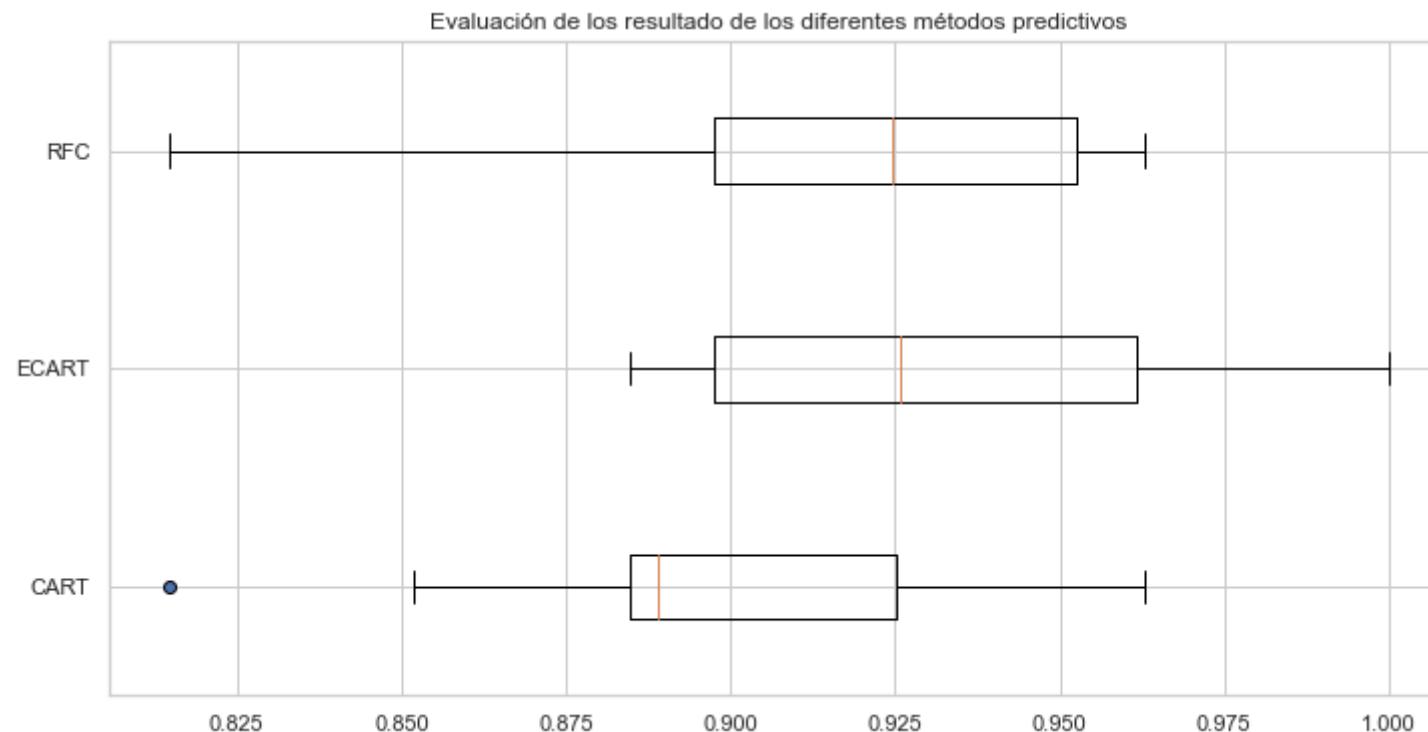
```
| Name: anomalia, dtype: int64
```

```
CART: Puntuaciones: 0.898718 - Desviación típica:(0.043857)
```

```
ECART: Puntuaciones: 0.932336 - Desviación típica:(0.036664)
```

```
RFC: Puntuaciones: 0.917664 - Desviación típica:(0.042815)
```

<Figure size 1080x360 with 0 Axes>



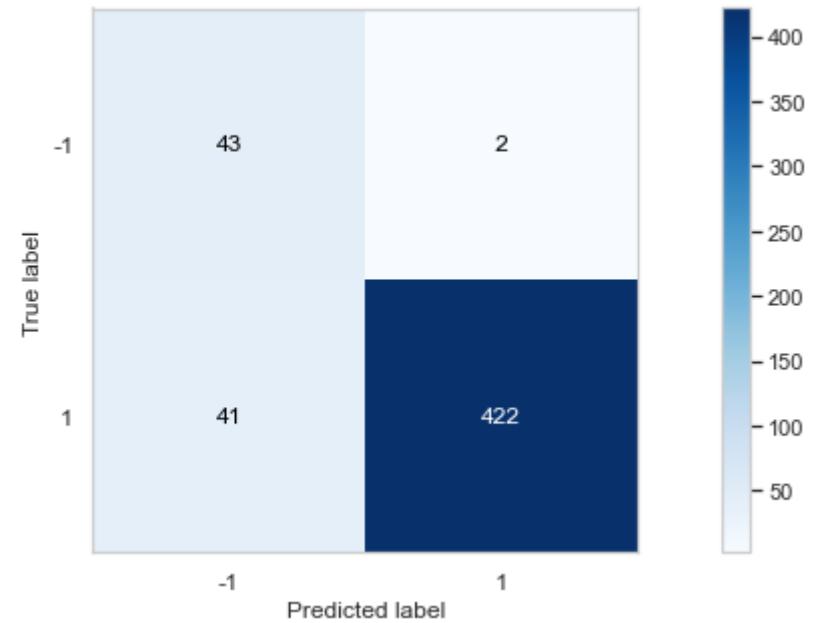
```

ExtraTreesClassifier(n_estimators=10, random_state=2021)
'accuracy_score'
0.9153543307086615
'f1_score'
0.95152198421646
'recall_score'
0.9114470842332614

```

|                                | Clase_0 | Clase_1 |
|--------------------------------|---------|---------|
| <b>TP</b>                      | 43.000  | 422.000 |
| <b>FP</b>                      | 41.000  | 2.000   |
| <b>FN</b>                      | 2.000   | 41.000  |
| <b>TN</b>                      | 422.000 | 43.000  |
| <b>Recall</b>                  | 0.956   | 0.911   |
| <b>Specificity</b>             | 0.911   | 0.956   |
| <b>PositivePredictiveValue</b> | 0.512   | 0.995   |
| <b>NegativePredictiveValue</b> | 0.995   | 0.512   |
| <b>FalsePositiveRate</b>       | 0.089   | 0.044   |
| <b>FalseNegativeRate</b>       | 0.044   | 0.089   |
| <b>Accuracy</b>                | 0.915   | 0.915   |
| <b>ErrorRate</b>               | 0.085   | 0.085   |

Confusion Matrix



### 10.1.1 Genera muestras sintéticas

In [164]:

```
# https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
from imblearn.over_sampling import SMOTE
from collections import Counter

# , test_x, , test_y

sm = SMOTE(random_state=semilla_aleatoria)
x_train, y_train = sm.fit_resample(train_x, train_y)

print('Conjunto de datos remuestreado %s' % Counter(y_train))
```

Conjunto de datos remuestreado Counter({1: 1899, -1: 1899})

```
In [165]: # Resultados cross_val_score / scoring="accuracy"
results = []
names= []
models_test(models_clf, x_train, y_train, cv_sample = cv_sel, jobs=2, model_scoring="accuracy", model_type="Classifier")

# Graficar Resultados
graphResults(results=results, labels=names)

display(upsampled_clf_model)
upsampled_clf_model.fit(x_train, y_train)
upsampled_predict = upsampled_clf_model.predict(test_x)

display("accuracy_score", accuracy_score(test_y, upsampled_predict))

display("f1_score", f1_score(test_y, upsampled_predict))

display("recall_score", recall_score(test_y, upsampled_predict))

upsampled_res = show_mc_mtr(test_y, upsampled_predict)

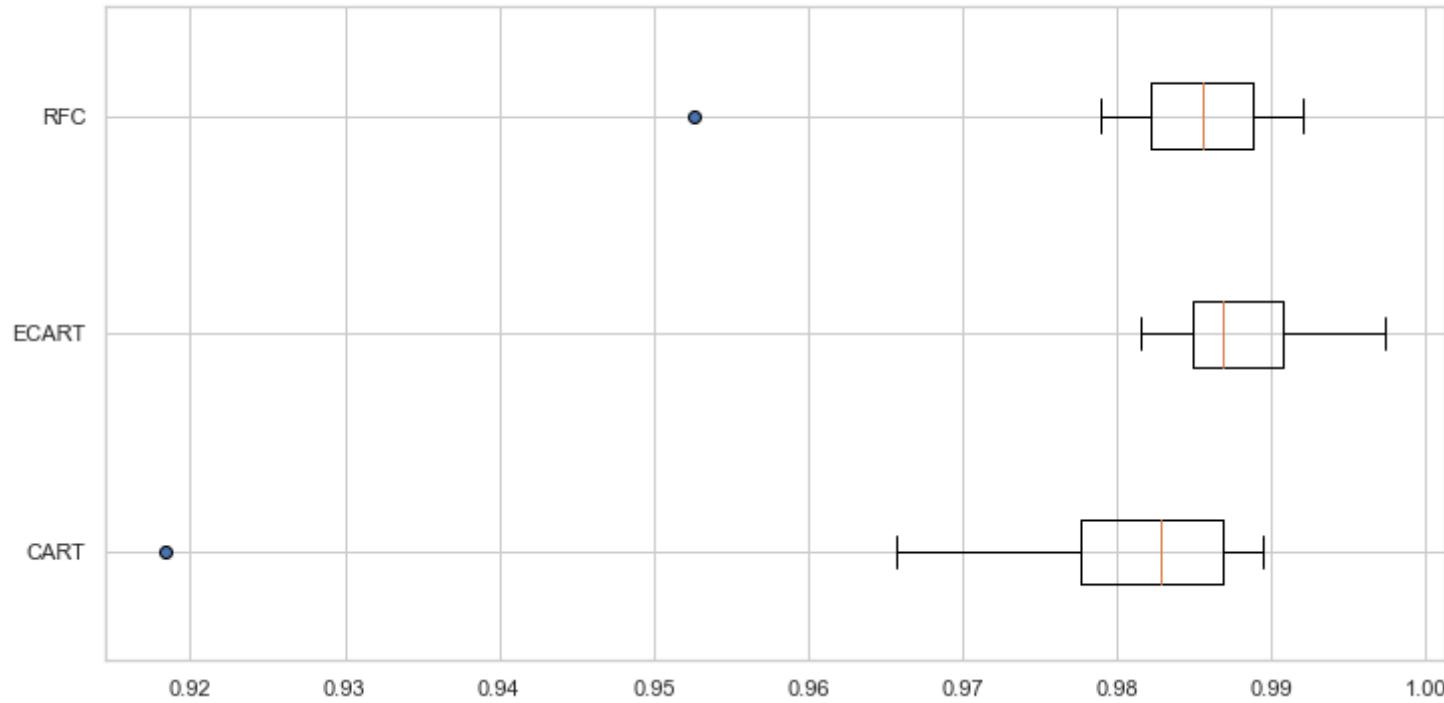
mmfg = pd.DataFrame(upsampled_res)
display(mmfg)

skplt.metrics.plot_confusion_matrix(test_y, upsampled_predict, normalize=False)
plt.show()
```

CART: Puntuaciones: 0.975781 - Desviación típica:(0.020203)  
ECART: Puntuaciones: 0.987888 - Desviación típica:(0.005025)  
RFC: Puntuaciones: 0.982624 - Desviación típica:(0.010670)

<Figure size 1080x360 with 0 Axes>

Evaluación de los resultado de los diferentes métodos predictivos



```
ExtraTreesClassifier(n_estimators=10, random_state=2021)
'accuracy_score'
0.9744094488188977
'f1_score'
0.9859762675296656
'recall_score'
0.9870410367170627
```

|  | Clase_0 | Clase_1 |
|--|---------|---------|
|--|---------|---------|

|    |        |         |
|----|--------|---------|
| TP | 38.000 | 457.000 |
|----|--------|---------|

|    |       |       |
|----|-------|-------|
| FP | 6.000 | 7.000 |
|----|-------|-------|

|                                | Clase_0 | Clase_1 |
|--------------------------------|---------|---------|
| <b>FN</b>                      | 7.000   | 6.000   |
| <b>TN</b>                      | 457.000 | 38.000  |
| <b>Recall</b>                  | 0.844   | 0.987   |
| <b>Specificity</b>             | 0.987   | 0.844   |
| <b>PositivePredictiveValue</b> | 0.864   | 0.985   |
| <b>NegativePredictiveValue</b> | 0.985   | 0.864   |
| <b>FalsePositiveRate</b>       | 0.013   | 0.156   |
| <b>FalseNegativeRate</b>       | 0.156   | 0.013   |
| <b>Accuracy</b>                | 0.974   | 0.974   |
| <b>ErrorRate</b>               | 0.026   | 0.026   |

