📌 Extracción

> ✅ Cargar los datos directamente desde la API utilizando Python.
>
> ✅ Convertir los datos a un DataFrame de Pandas para facilitar su manipulación.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px

import requests
import io
```

In [2]:
```
# Cargamos nuestro Github en una variable
url = 'https://raw.githubusercontent.com/WMRioga/Challenge_02_TelecomX/refs/heads/main/TelecomX_Data.json'

# Validamos nuestro documento json solo por comprobación
df = pd.read_json('./TelecomX_Data.json')
df
```

Out[2]:

	customerID	Churn	customer	phone	internet	account
0	0002-ORFBO	No	{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'One year', 'PaperlessBilling': '...
1	0003-MKNFE	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'Yes'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'Month-to-month', 'PaperlessBilli...
2	0004-TLHLJ	Yes	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'Fiber optic', 'OnlineSecu...	{'Contract': 'Month-to-month', 'PaperlessBilli...
3	0011-IGKFF	Yes	{'gender': 'Male', 'SeniorCitizen': 1, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'Fiber optic', 'OnlineSecu...	{'Contract': 'Month-to-month', 'PaperlessBilli...
4	0013-EXCHZ	Yes	{'gender': 'Female', 'SeniorCitizen': 1, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'Fiber optic', 'OnlineSecu...	{'Contract': 'Month-to-month', 'PaperlessBilli...
...	...	...	...	...	...	...
7262	9987-LUTYD	No	{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'One year', 'PaperlessBilling': '...
7263	9992-RRAMN	Yes	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'Yes'}	{'InternetService': 'Fiber optic', 'OnlineSecu...	{'Contract': 'Month-to-month', 'PaperlessBilli...
7264	9992-UJOEL	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'Month-to-month', 'PaperlessBilli...
7265	9993-LHIEB	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'Two year', 'PaperlessBilling': '...
7266	9995-HOTOH	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'No', 'MultipleLines': 'No ph...	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'Two year', 'PaperlessBilling': '...

7267 rows × 6 columns

In [3]:
```
# Descargamos el contenido de la URL directamente desde la nube para trabajar con el
response = requests.get(url)
content = response.content
df = pd.read_json(io.StringIO(content.decode('utf-8')))
df
```

Out[3]:

	customerID	Churn	customer	phone	internet	account
0	0002-ORFBO	No	{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'One year', 'PaperlessBilling': '...
1	0003-MKNFE	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'Yes'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'Month-to-month', 'PaperlessBilli...
2	0004-TLHLJ	Yes	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'Fiber optic', 'OnlineSecu...	{'Contract': 'Month-to-month', 'PaperlessBilli...
3	0011-IGKFF	Yes	{'gender': 'Male', 'SeniorCitizen': 1, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'Fiber optic', 'OnlineSecu...	{'Contract': 'Month-to-month', 'PaperlessBilli...
4	0013-EXCHZ	Yes	{'gender': 'Female', 'SeniorCitizen': 1, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'Fiber optic', 'OnlineSecu...	{'Contract': 'Month-to-month', 'PaperlessBilli...
...	...	...	...	...	...	...
7262	9987-LUTYD	No	{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'One year', 'PaperlessBilling': '...
7263	9992-RRAMN	Yes	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'Yes'}	{'InternetService': 'Fiber optic', 'OnlineSecu...	{'Contract': 'Month-to-month', 'PaperlessBilli...
7264	9992-UJOEL	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'Month-to-month', 'PaperlessBilli...
7265	9993-LHIEB	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'Two year', 'PaperlessBilling': '...
7266	9995-HOTOH	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'No', 'MultipleLines': 'No ph...	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'Two year', 'PaperlessBilling': '...

7267 rows × 6 columns

🔧 Transformación

✅ Explorar las columnas del dataset y verificar sus tipos de datos.

✅ Consultar el diccionario para comprender mejor el significado de las variables.

✅ Identificar las columnas más relevantes para el análisis de evasión.

In [4]:
```
# Validamos columnas
df.columns
```

Out[4]:
```
Index(['customerID', 'Churn', 'customer', 'phone', 'internet', 'account'], dtype='object')
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7267 entries, 0 to 7266
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   customerID  7267 non-null   object
 1   Churn       7267 non-null   object
 2   customer    7267 non-null   object
 3   phone       7267 non-null   object
 4   internet    7267 non-null   object
 5   account     7267 non-null   object
dtypes: object(6)
memory usage: 340.8+ KB
```

✅ Identificar las columnas para la normaliizacion de la base de datos.

In [6]:
```
# Separamos las columnas que no vamos normalizar
columnas_01 = df[['customerID','Churn']]
columnas_01
```

Out[6]:

	customerID	Churn
0	0002-ORFBO	No
1	0003-MKNFE	No
2	0004-TLHLJ	Yes
3	0011-IGKFF	Yes
4	0013-EXCHZ	Yes
...	...	...
7262	9987-LUTYD	No
7263	9992-RRAMN	Yes
7264	9992-UJOEL	No
7265	9993-LHIEB	No
7266	9995-HOTOH	No

7267 rows × 2 columns

In [7]:
```
# Tomamos las columnas que si vamos a normalizar del json
columnas_02 =df[['customer', 'phone', 'internet', 'account']]
columnas_02
```

Out[7]:

	customer	phone	internet	account
0	{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'One year', 'PaperlessBilling': '...
1	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'Yes'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'Month-to-month', 'PaperlessBilli...
2	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'Fiber optic', 'OnlineSecu...	{'Contract': 'Month-to-month', 'PaperlessBilli...
3	{'gender': 'Male', 'SeniorCitizen': 1, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'Fiber optic', 'OnlineSecu...	{'Contract': 'Month-to-month', 'PaperlessBilli...
4	{'gender': 'Female', 'SeniorCitizen': 1, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'Fiber optic', 'OnlineSecu...	{'Contract': 'Month-to-month', 'PaperlessBilli...
...	...	...	...	...
7262	{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'One year', 'PaperlessBilling': '...
7263	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'Yes'}	{'InternetService': 'Fiber optic', 'OnlineSecu...	{'Contract': 'Month-to-month', 'PaperlessBilli...
7264	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'Month-to-month', 'PaperlessBilli...
7265	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'Two year', 'PaperlessBilling': '...
7266	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'No', 'MultipleLines': 'No ph...	{'InternetService': 'DSL', 'OnlineSecurity': '...	{'Contract': 'Two year', 'PaperlessBilling': '...

7267 rows × 4 columns

In [8]:
```
# Crearemos un nuevo dataframe con las columnas que vamos a normalizar y las que no
df_merge = columnas_01
for i in columnas_02:
    column = pd.json_normalize(columnas_02[i])
    df_merge = pd.concat([df_merge, column], axis=1)
df_merge
```

Out[8]:

	customerID	Churn	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	Charges.Monthly	Cha
0	0002-ORFBO	No	Female	0	Yes	Yes	9	Yes	No	DSL	...	Yes	No	Yes	Yes	No	One year	Yes	Mailed check	65.60	
1	0003-MKNFE	No	Male	0	No	No	9	Yes	Yes	DSL	...	No	No	No	No	Yes	Month-to-month	No	Mailed check	59.90	
2	0004-TLHLJ	Yes	Male	0	No	No	4	Yes	No	Fiber optic	...	No	Yes	No	No	No	Month-to-month	Yes	Electronic check	73.90	
3	0011-IGKFF	Yes	Male	1	Yes	No	13	Yes	No	Fiber optic	...	Yes	Yes	No	Yes	Yes	Month-to-month	Yes	Electronic check	98.00	
4	0013-EXCHZ	Yes	Female	1	Yes	No	3	Yes	No	Fiber optic	...	No	No	Yes	Yes	No	Month-to-month	Yes	Mailed check	83.90	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7262	9987-LUTYD	No	Female	0	No	No	13	Yes	No	DSL	...	No	No	Yes	No	No	One year	No	Mailed check	55.15	
7263	9992-RRAMN	Yes	Male	0	Yes	No	22	Yes	Yes	Fiber optic	...	No	No	No	No	Yes	Month-to-month	Yes	Electronic check	85.10	
7264	9992-UJOEL	No	Male	0	No	No	2	Yes	No	DSL	...	Yes	No	No	No	No	Month-to-month	Yes	Mailed check	50.30	
7265	9993-LHIEB	No	Male	0	Yes	Yes	67	Yes	No	DSL	...	No	Yes	Yes	No	Yes	Two year	No	Mailed check	67.85	
7266	9995-HOTOH	No	Male	0	Yes	Yes	63	No	No phone service	DSL	...	Yes	Yes	No	Yes	Yes	Two year	No	Electronic check	59.00	

7267 rows × 21 columns

In [9]:
```
df_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7267 entries, 0 to 7266
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7267 non-null   object 
 1   Churn             7267 non-null   object 
 2   gender            7267 non-null   object 
 3   SeniorCitizen     7267 non-null   int64  
 4   Partner           7267 non-null   object 
 5   Dependents        7267 non-null   object 
 6   tenure            7267 non-null   int64  
 7   PhoneService      7267 non-null   object 
 8   MultipleLines     7267 non-null   object 
 9   InternetService   7267 non-null   object 
 10  OnlineSecurity    7267 non-null   object 
 11  OnlineBackup      7267 non-null   object 
 12  DeviceProtection  7267 non-null   object 
 13  TechSupport       7267 non-null   object 
 14  StreamingTV       7267 non-null   object 
 15  StreamingMovies   7267 non-null   object 
 16  Contract          7267 non-null   object 
 17  PaperlessBilling  7267 non-null   object 
 18  PaymentMethod     7267 non-null   object 
 19  Charges.Monthly   7267 non-null   float64
 20  Charges.Total     7267 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.2+ MB
```

In [10]:
```
df_merge.columns
```

Out[10]:
```
Index(['customerID', 'Churn', 'gender', 'SeniorCitizen', 'Partner',
       'Dependents', 'tenure', 'PhoneService', 'MultipleLines',
       'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'Charges.Monthly',
       'Charges.Total'],
      dtype='object')
```

✅ Validar si hay problemas en los datos que puedan afectar el análisis.

✅ Prestar atención a valores ausentes, duplicados, errores de formato e inconsistencias en las categorías.

In [11]:
```
# Creamos una columna de validacion de datos, donde analicemos que datos son problematicos usando el metodo isna() para saber si son nulos o no
no_numericos = pd.to_numeric(df_merge['Charges.Total'], errors='coerce').isna()
print(no_numericos.value_counts())
# Mostramos los valores únicos que están causando el problema
print("Valores problemáticos encontrados:")
print(df_merge.loc[no_numericos, 'Charges.Total'].unique())
```

```
Charges.Total
False    7256
True       11
Name: count, dtype: int64
Valores problemáticos encontrados:
[' ']
```

In [12]:
```
# Como validamos que los valores problematicos son valores vacios, los reemplazamos por 0
# Convertimos a numérico (los espacios se vuelven NaN) y luego llenamos esos NaN con 0
df_merge['Charges.Total'] = pd.to_numeric(df_merge['Charges.Total'], errors='coerce').fillna(0)

# Nos aseguramos que el tipo de dato sea float
df_merge['Charges.Total'] = df_merge['Charges.Total'].astype(float)
df_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7267 entries, 0 to 7266
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7267 non-null   object 
 1   Churn             7267 non-null   object 
 2   gender            7267 non-null   object 
 3   SeniorCitizen     7267 non-null   int64  
 4   Partner           7267 non-null   object 
 5   Dependents        7267 non-null   object 
 6   tenure            7267 non-null   int64  
 7   PhoneService      7267 non-null   object 
 8   MultipleLines     7267 non-null   object 
 9   InternetService   7267 non-null   object 
 10  OnlineSecurity    7267 non-null   object 
 11  OnlineBackup      7267 non-null   object 
 12  DeviceProtection  7267 non-null   object 
 13  TechSupport       7267 non-null   object 
 14  StreamingTV       7267 non-null   object 
 15  StreamingMovies   7267 non-null   object 
 16  Contract          7267 non-null   object 
 17  PaperlessBilling  7267 non-null   object 
 18  PaymentMethod     7267 non-null   object 
 19  Charges.Monthly   7267 non-null   float64
 20  Charges.Total     7267 non-null   float64
dtypes: float64(2), int64(2), object(17)
memory usage: 1.2+ MB
```

✅ Identificar las inconsistencias y aplicar las correcciones necesarias. Ajustar los datos para asegurar de que estén completos y coherentes, preparándolos para las siguientes etapas del análisis.

In [13]:
```
# Como hemos observado muchas columnas con YES y NO, las vamos a convertir en booleanas
columnas_booleanas = ['Churn','Partner','Dependents', 'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling']
columnas_booleanas
```

Out[13]:
```
['Churn',
 'Partner',
 'Dependents',
 'PhoneService',
 'MultipleLines',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'PaperlessBilling']
```

In [14]:
```
# Necesitamos ver que de verdad solo existan valores YES o NO en las columnas, por lo que creare un for para verlos
for i in columnas_booleanas:
    print([i],df_merge[i].unique())
```

```
['Churn'] ['No' 'Yes' '']
['Partner'] ['Yes' 'No']
['Dependents'] ['Yes' 'No']
['PhoneService'] ['Yes' 'No']
['MultipleLines'] ['No' 'Yes' 'No phone service']
['OnlineSecurity'] ['No' 'Yes' 'No internet service']
['OnlineBackup'] ['Yes' 'No' 'No internet service']
['DeviceProtection'] ['No' 'Yes' 'No internet service']
['TechSupport'] ['Yes' 'No' 'No internet service']
['StreamingTV'] ['Yes' 'No' 'No internet service']
['StreamingMovies'] ['No' 'Yes' 'No internet service']
['PaperlessBilling'] ['Yes' 'No']
```

In [15]:
```
# Como el objeto del Challenge es analizar el comportamiento de la columna Churn,
# al observar campos vacios (''), vamos a validar primero que tienen las columnas que nos interesan

df_merge.query('Churn == ""')
```

Out[15]:

	customerID	Churn	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	Charges.Monthly	Cha
30	0047-ZHDTW		Female	0	No	No	11	Yes	Yes	Fiber optic	...	No	No	No	No	No	Month-to-month	Yes	Bank transfer (automatic)	79.00	
75	0120-YZLQA		Male	0	No	No	71	Yes	No	No	...	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	Yes	Credit card (automatic)	19.90	
96	0154-QYHJU		Male	0	No	No	29	Yes	No	DSL	...	Yes	No	Yes	No	No	One year	Yes	Electronic check	58.75	
98	0162-RZGMZ		Female	1	No	No	5	Yes	No	DSL	...	Yes	No	Yes	No	No	Month-to-month	No	Credit card (automatic)	59.90	
175	0274-VVQOQ		Male	1	Yes	No	65	Yes	Yes	Fiber optic	...	Yes	Yes	No	Yes	Yes	One year	Yes	Bank transfer (automatic)	103.15	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7158	9840-GSRFX		Female	0	No	No	14	Yes	Yes	DSL	...	Yes	No	No	No	No	One year	Yes	Mailed check	54.25	
7180	9872-RZQQB		Female	0	Yes	No	49	No	No phone service	DSL	...	No	No	No	Yes	No	Month-to-month	No	Bank transfer (automatic)	40.65	
7211	9920-GNDMB		Male	0	No	No	9	Yes	Yes	Fiber optic	...	No	No	No	No	No	Month-to-month	Yes	Electronic check	76.25	
7239	9955-RVWSC		Female	0	Yes	Yes	67	Yes	No	No	...	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	Yes	Bank transfer (automatic)	19.25	
7247	9966-VYRTZ		Female	0	Yes	Yes	31	Yes	No	No	...	No internet service	No internet service	No internet service	No internet service	No internet service	Month-to-month	Yes	Mailed check	19.55	

224 rows × 21 columns

In [16]:
```
df_merge.query('Churn == "No"').sample(5)
```

Out[16]:

	customerID	Churn	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	Charges.Monthly	Cha
611	0871-OPBXW	No	Female	0	No	No	2	Yes	No	No	...	No internet service	No internet service	No internet service	No internet service	No internet service	Month-to-month	Yes	Mailed check	20.05	
5027	6873-UDNLD	No	Male	0	No	No	40	Yes	No	DSL	...	No	Yes	No	No	Yes	Month-to-month	No	Electronic check	67.45	
826	1171-TYKUR	No	Male	0	Yes	No	47	Yes	Yes	No	...	No internet service	No internet service	No internet service	No internet service	No internet service	Month-to-month	No	Electronic check	25.40	
3287	4583-PARNH	No	Male	1	Yes	No	16	Yes	No	Fiber optic	...	No	Yes	Yes	No	Yes	Month-to-month	Yes	Electronic check	91.55	
6354	8739-QOTTN	No	Female	0	Yes	No	2	Yes	No	No	...	No internet service	No internet service	No internet service	No internet service	No internet service	Month-to-month	No	Mailed check	20.35	

5 rows × 21 columns

In [17]:
```
df_merge.query('Churn == "Yes"').sample(5)
```

Out[17]:

	customerID	Churn	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	Charges.Monthly	Cha
4077	5577-OTWWW	Yes	Female	0	No	No	15	Yes	No	No	...	No internet service	No internet service	No internet service	No internet service	No internet service	Month-to-month	No	Bank transfer (automatic)	19.75	
720	1031-IIDEO	Yes	Female	0	No	No	1	Yes	No	Fiber optic	...	No	No	No	No	No	Month-to-month	Yes	Electronic check	70.85	
7128	9809-IMGCQ	Yes	Male	1	No	No	22	Yes	Yes	Fiber optic	...	No	No	No	Yes	Yes	Month-to-month	Yes	Electronic check	96.70	
2568	3583-EKAPL	Yes	Male	0	No	No	1	Yes	No	DSL	...	No	No	No	No	Yes	Month-to-month	Yes	Electronic check	55.00	
6310	8679-JOEVF	Yes	Female	1	No	No	16	Yes	No	DSL	...	No	No	Yes	Yes	No	Month-to-month	No	Electronic check	59.40	

5 rows × 21 columns

In [18]:
```
# Al comprar los valores vacios de la columna Churn, con el fin de no afectar futuros analisis,
# solo los dropearemos de la base de datos los valores que estan vacios (''). esto no afectara a la base de datos ni negativa ni positivamente.

df_merge_clean = df_merge[df_merge['Churn'] != ''].copy()
df_merge_clean
```

Out[18]:

	customerID	Churn	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	Charges.Monthly	Cha
0	0002-ORFBO	No	Female	0	Yes	Yes	9	Yes	No	DSL	...	Yes	No	Yes	Yes	No	One year	Yes	Mailed check	65.60	
1	0003-MKNFE	No	Male	0	No	No	9	Yes	Yes	DSL	...	No	No	No	No	Yes	Month-to-month	No	Mailed check	59.90	
2	0004-TLHLJ	Yes	Male	0	No	No	4	Yes	No	Fiber optic	...	No	Yes	No	No	No	Month-to-month	Yes	Electronic check	73.90	
3	0011-IGKFF	Yes	Male	1	Yes	No	13	Yes	No	Fiber optic	...	Yes	Yes	No	Yes	Yes	Month-to-month	Yes	Electronic check	98.00	
4	0013-EXCHZ	Yes	Female	1	Yes	No	3	Yes	No	Fiber optic	...	No	No	Yes	Yes	No	Month-to-month	Yes	Mailed check	83.90	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7262	9987-LUTYD	No	Female	0	No	No	13	Yes	No	DSL	...	No	No	Yes	No	No	One year	No	Mailed check	55.15	
7263	9992-RRAMN	Yes	Male	0	Yes	No	22	Yes	Yes	Fiber optic	...	No	No	No	No	Yes	Month-to-month	Yes	Electronic check	85.10	
7264	9992-UJOEL	No	Male	0	No	No	2	Yes	No	DSL	...	Yes	No	No	No	No	Month-to-month	Yes	Mailed check	50.30	
7265	9993-LHIEB	No	Male	0	Yes	Yes	67	Yes	No	DSL	...	No	Yes	Yes	No	Yes	Two year	No	Mailed check	67.85	
7266	9995-HOTOH	No	Male	0	Yes	Yes	63	No	No phone service	DSL	...	Yes	Yes	No	Yes	Yes	Two year	No	Electronic check	59.00	

7043 rows × 21 columns

In [19]:
```
for i in columnas_booleanas:
    print([i],df_merge_clean[i].unique())
```

```
['Churn'] ['No' 'Yes']
['Partner'] ['Yes' 'No']
['Dependents'] ['Yes' 'No']
['PhoneService'] ['Yes' 'No']
['MultipleLines'] ['No' 'Yes' 'No phone service']
['OnlineSecurity'] ['No' 'Yes' 'No internet service']
['OnlineBackup'] ['Yes' 'No' 'No internet service']
['DeviceProtection'] ['No' 'Yes' 'No internet service']
['TechSupport'] ['Yes' 'No' 'No internet service']
['StreamingTV'] ['Yes' 'No' 'No internet service']
['StreamingMovies'] ['No' 'Yes' 'No internet service']
['PaperlessBilling'] ['Yes' 'No']
```

In [20]:
```
# Ahora, las columnas con valores diferentes a Yes o No, las vamos a cambiar No

for i in columnas_booleanas:
    df_merge_clean[i] = df_merge_clean[i].replace(['No phone service'], 'No')
    df_merge_clean[i] = df_merge_clean[i].replace(['No internet service'], 'No')
```

In [21]:
```
for i in columnas_booleanas:
    print([i],df_merge_clean[i].unique())
```

```
['Churn'] ['No' 'Yes']
['Partner'] ['Yes' 'No']
['Dependents'] ['Yes' 'No']
['PhoneService'] ['Yes' 'No']
['MultipleLines'] ['No' 'Yes']
['OnlineSecurity'] ['No' 'Yes']
['OnlineBackup'] ['Yes' 'No']
['DeviceProtection'] ['No' 'Yes']
['TechSupport'] ['Yes' 'No']
['StreamingTV'] ['Yes' 'No']
['StreamingMovies'] ['No' 'Yes']
['PaperlessBilling'] ['Yes' 'No']
```

✅ La estandarización y transformación de datos. Hacer que la información sea más consistente, comprensible y adecuada para el análisis. Durante esta fase, se deven convertir valores textuales como "Sí" y "No" en valores binarios (1 y 0), lo que facilitará el procesamiento matemático y la aplicación de modelos analíticos.

In [22]:
```
# Una vez validadas las columnas que solo sean Yes y No, valos a volverlas en booleanas
# Definimos el mapeo
mapeo = {'Yes': True, 'No': False}

# Aplicamos el cambio a todo el bloque
for col in columnas_booleanas:
    # .str.strip() elimina espacios invisibles antes de mapear
    df_merge_clean[col] = df_merge_clean[col].astype(str).str.strip().map(mapeo)

df_merge_clean.sample(10)
```

Out[22]:

	customerID	Churn	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	Charges.Monthly	Cha
2080	2927-CVULT	False	Female	0	True	True	53	True	True	Fiber optic	...	True	True	True	False	True	Two year	True	Bank transfer (automatic)	104.05	
215	0325-XBFAC	True	Male	0	False	False	8	True	False	Fiber optic	...	False	True	False	True	True	Month-to-month	True	Electronic check	94.70	
4237	5828-DWPIL	False	Male	1	True	False	62	True	True	Fiber optic	...	True	True	False	False	False	Month-to-month	True	Electronic check	89.10	
2487	3470-OBUET	False	Female	0	True	True	67	True	True	DSL	...	True	False	True	True	False	Two year	False	Credit card (automatic)	74.00	
5285	7219-TLZHO	False	Female	0	True	True	4	True	False	No	...	False	False	False	False	False	Month-to-month	False	Mailed check	20.85	
2732	3780-YVMFA	False	Female	0	True	True	8	True	False	DSL	...	False	False	True	True	True	Month-to-month	True	Electronic check	68.55	
1171	1670-SVOWZ	True	Female	0	True	True	14	True	False	Fiber optic	...	True	False	True	False	True	Month-to-month	True	Credit card (automatic)	89.65	
986	1389-CXMLU	True	Male	1	False	False	3	True	True	Fiber optic	...	False	True	False	True	False	Month-to-month	False	Electronic check	91.05	
6163	8436-BJUMM	True	Male	0	True	True	26	True	True	Fiber optic	...	False	False	False	True	False	Month-to-month	True	Electronic check	83.75	
5798	7928-VJYAB	False	Male	0	True	True	11	True	False	Fiber optic	...	False	False	False	True	True	Month-to-month	False	Electronic check	90.60	

10 rows × 21 columns

In [23]:
```
# Debido a que la columna SeniorCitizen es de tipo int, vamos a convertirla a booleana
df_merge_clean['SeniorCitizen'].value_counts()
```

Out[23]:
```
SeniorCitizen
0    5901
1    1142
Name: count, dtype: int64
```

In [24]:
```
df_merge_clean['SeniorCitizen'] = df_merge_clean['SeniorCitizen'].astype(bool)
df_merge_clean['SeniorCitizen'].value_counts()
```

Out[24]:
```
SeniorCitizen
False    5901
True     1142
Name: count, dtype: int64
```

In [25]:
```
df_merge_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7043 entries, 0 to 7266
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   Churn             7043 non-null   bool
 2   gender            7043 non-null   object
 3   SeniorCitizen     7043 non-null   bool
 4   Partner           7043 non-null   bool
 5   Dependents        7043 non-null   bool
 6   tenure            7043 non-null   int64
 7   PhoneService      7043 non-null   bool
 8   MultipleLines     7043 non-null   bool
 9   InternetService   7043 non-null   object
 10  OnlineSecurity    7043 non-null   bool
 11  OnlineBackup      7043 non-null   bool
 12  DeviceProtection  7043 non-null   bool
 13  TechSupport       7043 non-null   bool
 14  StreamingTV       7043 non-null   bool
 15  StreamingMovies   7043 non-null   bool
 16  Contract          7043 non-null   object
 17  PaperlessBilling  7043 non-null   bool
 18  PaymentMethod     7043 non-null   object
 19  Charges.Monthly   7043 non-null   float64
 20  Charges.Total     7043 non-null   float64
dtypes: bool(13), float64(2), int64(1), object(5)
memory usage: 584.6+ KB
```

✅ Renombrar columnas y datos hace que la información sea más accesible y fácil de entender, especialmente cuando se trabaja con fuentes externas o términos técnicos.

In [26]:
```
# Cambiamos el formato de texto de los headers de las columnas Charges.Monthly y Charges.Total, para que queden como los demas:
# Charges.Monthly to ChargesMonthly
# Charges.Total to ChargesTotal
df_merge_clean.rename(columns={'Charges.Monthly': 'ChargesMonthly', 'Charges.Total': 'ChargesTotal'}, inplace=True)
df_merge_clean.columns
```

Out[26]:
```
Index(['customerID', 'Churn', 'gender', 'SeniorCitizen', 'Partner',
       'Dependents', 'tenure', 'PhoneService', 'MultipleLines',
       'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'ChargesMonthly', 'ChargesTotal'],
      dtype='object')
```

In [27]:
```
df_merge_clean.sample(10)
```

Out[27]:

	customerID	Churn	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	ChargesMonthly	Char
476	0674-GCDXG	False	Male	False	False	False	63	True	True	DSL	...	False	True	True	False	True	Two year	True	Bank transfer (automatic)	71.90	
2065	2907-ILJBN	False	Female	False	True	True	11	True	False	No	...	False	False	False	False	False	One year	False	Mailed check	20.60	
4496	6166-YIPFO	False	Male	False	True	False	72	False	False	DSL	...	True	True	True	True	True	Two year	True	Electronic check	64.70	
4377	5996-NRVXR	False	Male	True	True	False	40	True	False	Fiber optic	...	True	True	True	True	False	One year	True	Credit card (automatic)	98.15	
1186	1697-BCSHV	False	Female	False	True	True	58	True	True	DSL	...	True	False	False	True	False	Month-to-month	True	Bank transfer (automatic)	66.80	
4270	5872-OEQNH	False	Female	False	False	False	60	False	False	DSL	...	False	True	False	False	True	One year	True	Electronic check	44.45	
4003	5474-LAMUQ	False	Male	False	True	False	24	True	False	No	...	False	False	False	False	False	One year	False	Mailed check	20.10	
3352	4659-NZRUF	True	Female	False	False	False	19	True	True	Fiber optic	...	True	False	False	False	True	Month-to-month	True	Electronic check	95.15	
5980	8174-LNWMW	False	Female	False	False	False	31	True	False	No	...	False	False	False	False	False	Two year	False	Credit card (automatic)	20.90	
4088	5599-HVLTW	False	Female	True	False	False	14	True	True	Fiber optic	...	False	True	False	False	False	Month-to-month	True	Bank transfer (automatic)	80.35	

10 rows × 21 columns

In [28]:
```
df_merge_clean['gender'].value_counts()
```

Out[28]:
```
gender
Male      3555
Female    3488
Name: count, dtype: int64
```

In [29]:
```
df_merge_clean['InternetService'].value_counts()
```

Out[29]:
```
InternetService
Fiber optic    3096
DSL            2421
No             1526
Name: count, dtype: int64
```

In [30]:
```
df_merge_clean['Contract'].value_counts()
```

Out[30]:
```
Contract
Month-to-month    3875
Two year          1695
One year          1473
Name: count, dtype: int64
```

In [31]:
```
df_merge_clean['PaymentMethod'].value_counts()
```

Out[31]:
```
PaymentMethod
Electronic check             2365
Mailed check                 1612
Bank transfer (automatic)    1544
Credit card (automatic)      1522
Name: count, dtype: int64
```

In [32]:
```
df_merge_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7043 entries, 0 to 7266
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   Churn             7043 non-null   bool
 2   gender            7043 non-null   object
 3   SeniorCitizen     7043 non-null   bool
 4   Partner           7043 non-null   bool
 5   Dependents        7043 non-null   bool
 6   tenure            7043 non-null   int64
 7   PhoneService      7043 non-null   bool
 8   MultipleLines     7043 non-null   bool
 9   InternetService   7043 non-null   object
 10  OnlineSecurity    7043 non-null   bool
 11  OnlineBackup      7043 non-null   bool
 12  DeviceProtection  7043 non-null   bool
 13  TechSupport       7043 non-null   bool
 14  StreamingTV       7043 non-null   bool
 15  StreamingMovies   7043 non-null   bool
 16  Contract          7043 non-null   object
 17  PaperlessBilling  7043 non-null   bool
 18  PaymentMethod     7043 non-null   object
 19  ChargesMonthly    7043 non-null   float64
 20  ChargesTotal      7043 non-null   float64
dtypes: bool(13), float64(2), int64(1), object(5)
memory usage: 584.6+ KB
```

📌 Crear la columna "Cuentas_Diarias". Utiliza la facturación mensual para calcular el valor diario, proporcionando una visión más detallada del comportamiento de los clientes a lo largo del tiempo.

In [33]:
```
# Ahora que los datos están limpios, es momento de crear la columna "Cuentas_Diarias".
# Utilizaremos la facturación mensual para calcular el valor diario, proporcionando una visión más detallada del comportamiento de los clientes a lo largo del tiempo.
# Usaremos 30 dias/mes
df_merge_clean['Cuentas_Diarias'] = (df_merge_clean['ChargesMonthly']/30).round(2)
df_merge_clean
```

Out[33]:

	customerID	Churn	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	ChargesMonthly	ChargesTotal	Cuen
0	0002-ORFBO	False	Female	False	True	True	9	True	False	DSL	...	False	True	True	False	One year	True	Mailed check	65.60	593.30	
1	0003-MKNFE	False	Male	False	False	False	9	True	True	DSL	...	False	False	False	True	Month-to-month	False	Mailed check	59.90	542.40	
2	0004-TLHLJ	True	Male	False	False	False	4	True	False	Fiber optic	...	True	False	False	False	Month-to-month	True	Electronic check	73.90	280.85	
3	0011-IGKFF	True	Male	True	True	False	13	True	False	Fiber optic	...	True	False	True	True	Month-to-month	True	Electronic check	98.00	1237.85	
4	0013-EXCHZ	True	Female	True	True	False	3	True	False	Fiber optic	...	False	True	True	False	Month-to-month	True	Mailed check	83.90	267.40	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7262	9987-LUTYD	False	Female	False	False	False	13	True	False	DSL	...	False	True	False	False	One year	False	Mailed check	55.15	742.90	
7263	9992-RRAMN	True	Male	False	True	False	22	True	True	Fiber optic	...	False	False	False	True	Month-to-month	True	Electronic check	85.10	1873.70	
7264	9992-UJOEL	False	Male	False	False	False	2	True	False	DSL	...	False	False	False	False	Month-to-month	True	Mailed check	50.30	92.75	
7265	9993-LHIEB	False	Male	False	True	True	67	True	False	DSL	...	True	True	False	True	Two year	False	Mailed check	67.85	4627.65	
7266	9995-HOTOH	False	Male	False	True	True	63	False	False	DSL	...	True	False	True	True	Two year	False	Electronic check	59.00	3707.60	

7043 rows × 22 columns

In [34]:
```
df_merge_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7043 entries, 0 to 7266
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   Churn             7043 non-null   bool   
 2   gender            7043 non-null   object 
 3   SeniorCitizen     7043 non-null   bool   
 4   Partner           7043 non-null   bool   
 5   Dependents        7043 non-null   bool   
 6   tenure            7043 non-null   int64  
 7   PhoneService      7043 non-null   bool   
 8   MultipleLines     7043 non-null   bool   
 9   InternetService   7043 non-null   object 
 10  OnlineSecurity    7043 non-null   bool   
 11  OnlineBackup      7043 non-null   bool   
 12  DeviceProtection  7043 non-null   bool   
 13  TechSupport       7043 non-null   bool   
 14  StreamingTV       7043 non-null   bool   
 15  StreamingMovies   7043 non-null   bool   
 16  Contract          7043 non-null   object 
 17  PaperlessBilling  7043 non-null   bool   
 18  PaymentMethod     7043 non-null   object 
 19  ChargesMonthly    7043 non-null   float64
 20  ChargesTotal      7043 non-null   float64
 21  Cuentas_Diarias   7043 non-null   float64
dtypes: bool(13), float64(3), int64(1), object(5)
memory usage: 639.6+ KB
```

📊 Carga y análisis

> Realiza un análisis descriptivo de los datos, calculando métricas como media, mediana, desviación estándar y otras medidas que ayuden a comprender mejor la distribución y el comportamiento de los clientes.

In [35]:
```
df_merge_clean.sample(10)
```

Out[35]:

	customerID	Churn	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	ChargesMonthly	ChargesTotal	Cuen
1356	1960-UYCNN	False	Male	False	False	False	10	True	True	Fiber optic	...	True	False	False	True	Month-to-month	True	Electronic check	95.25	1021.55	
3442	4770-QAZXN	False	Female	False	False	False	13	True	False	No	...	False	False	False	False	Month-to-month	False	Credit card (automatic)	19.45	232.10	
3236	4521-YEEHE	False	Female	False	True	False	18	True	True	Fiber optic	...	False	True	True	False	Month-to-month	True	Electronic check	88.85	1594.75	
1509	2171-UDMFD	False	Male	False	True	True	32	True	False	No	...	False	False	False	False	Month-to-month	True	Credit card (automatic)	19.45	674.55	
4722	6474-FVJLC	True	Male	False	False	False	2	True	True	Fiber optic	...	False	False	False	True	Month-to-month	True	Electronic check	86.00	165.45	
3210	4482-EWFMI	False	Female	False	False	False	2	True	False	Fiber optic	...	False	False	False	False	Month-to-month	True	Electronic check	69.70	135.20	
113	0193-ESZXP	True	Female	True	True	False	58	True	False	Fiber optic	...	False	True	True	True	One year	True	Credit card (automatic)	105.50	6205.50	
3757	5167-GBFRE	True	Male	True	False	False	4	False	False	DSL	...	False	False	False	False	Month-to-month	True	Bank transfer (automatic)	25.20	102.50	
1252	1792-UXAFY	True	Female	True	False	False	17	True	False	Fiber optic	...	True	False	True	False	Month-to-month	True	Electronic check	89.15	1496.90	
5527	7576-ASEJU	False	Female	False	True	True	41	True	False	DSL	...	False	True	True	True	One year	True	Credit card (automatic)	74.70	3187.65	

10 rows × 22 columns

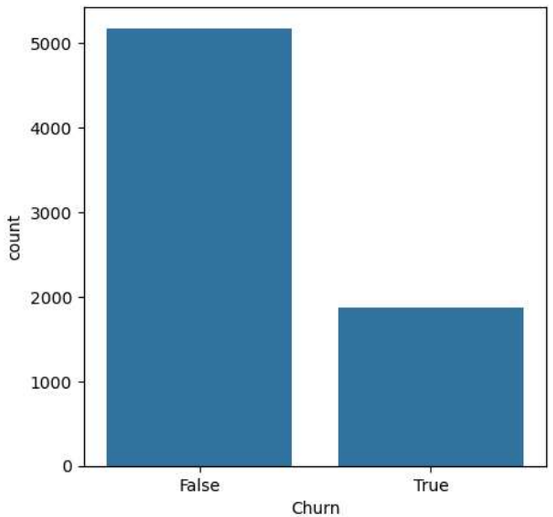
In [36]:
```
df_merge_clean.describe()
```

Out[36]:

	tenure	ChargesMonthly	ChargesTotal	Cuentas_Diarias
count	7043.000000	7043.000000	7043.000000	7043.000000
mean	32.371149	64.761692	2279.734304	2.158675
std	24.559481	30.090047	2266.794470	1.003088
min	0.000000	18.250000	0.000000	0.610000
25%	9.000000	35.500000	398.550000	1.180000
50%	29.000000	70.350000	1394.550000	2.340000
75%	55.000000	89.850000	3786.600000	2.990000
max	72.000000	118.750000	8684.800000	3.960000

📊 Comprender cómo está distribuida la variable "churn" (evasión) entre los clientes.

In [37]:
```
import seaborn as sns

fig, ax = plt.subplots(figsize=(5, 5))

sns.countplot(data=df_merge_clean, x='Churn')
plt.show()
```



In [63]:
```
df_merge_clean['Churn'].value_counts(normalize=True).round(6)*100
```

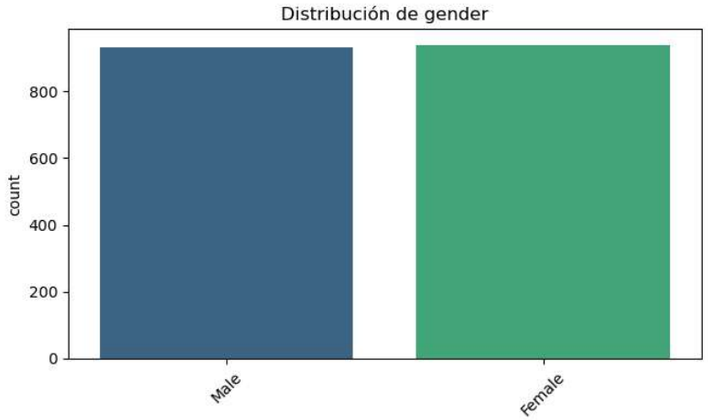
Out[63]:
```
Churn
False    73.463
True     26.537
Name: proportion, dtype: float64
```

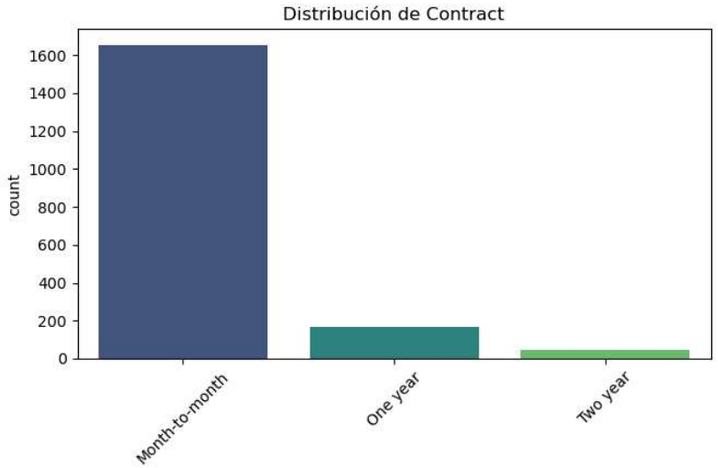
📊 Cómo se distribuye la evasión según variables categóricas, como **género, tipo de contrato, método de pago,** entre otras.

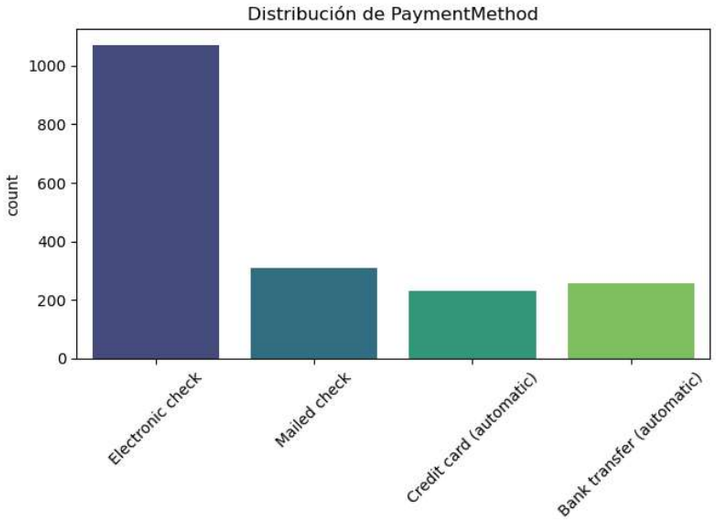
In [38]:
```
churn_true = df_merge_clean[df_merge_clean['Churn'] == True]
churn_false = df_merge_clean[df_merge_clean['Churn'] == False]
evacion = ['gender','Contract','PaymentMethod']
evacion
```

Out[38]:
```
['gender', 'Contract', 'PaymentMethod']
```

In [39]:
```
fig, ax = plt.subplots(1, 3, figsize=(20, 5))

# Iteramos para crear cada barplot (count plot)
for i, col in enumerate(evacion):
    sns.countplot(data=churn_true, x=col, hue=col, ax=ax[i], palette='viridis', legend=False)
    ax[i].set_title(f'Distribución de {col}')
    ax[i].set_xlabel('')
    # Rotamos las etiquetas del eje X
    ax[i].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```







In [64]:
```
for i in evacion:
    print(df_merge_clean[i].value_counts(normalize=True).round(6)*100)
```

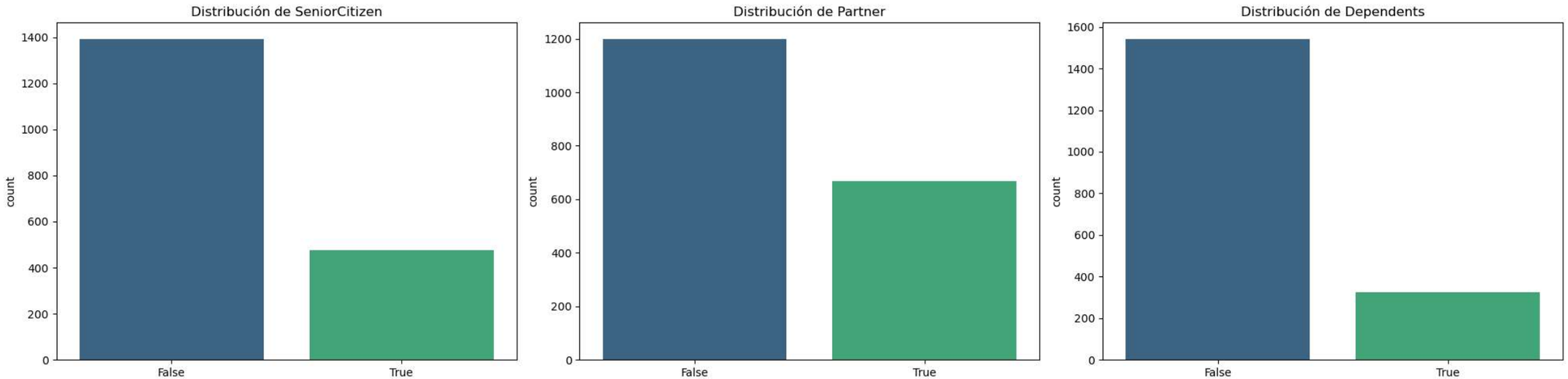
```
gender
Male      50.4756
Female    49.5244
Name: proportion, dtype: float64
Contract
Month-to-month    55.0192
Two year          24.0664
One year          20.9144
Name: proportion, dtype: float64
PaymentMethod
Electronic check             33.5794
Mailed check                 22.8880
Bank transfer (automatic)    21.9225
Credit card (automatic)      21.6101
Name: proportion, dtype: float64
```

In [40]:
```
df_merge_clean.columns
```

Out[40]:
```
Index(['customerID', 'Churn', 'gender', 'SeniorCitizen', 'Partner',
       'Dependents', 'tenure', 'PhoneService', 'MultipleLines',
       'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'ChargesMonthly', 'ChargesTotal',
       'Cuentas_Diarias'],
      dtype='object')
```

In [41]:
```
analizys_familia = ['SeniorCitizen', 'Partner', 'Dependents']

fig, ax = plt.subplots(1, 3, figsize=(20, 5))

# Iteramos para crear cada barplot (count plot)
for i, col in enumerate(analizys_familia):
    sns.countplot(data=churn_true, x=col, hue=col, ax=ax[i], palette='viridis', legend=False)
    ax[i].set_title(f'Distribución de {col}')
    ax[i].set_xlabel('')
```

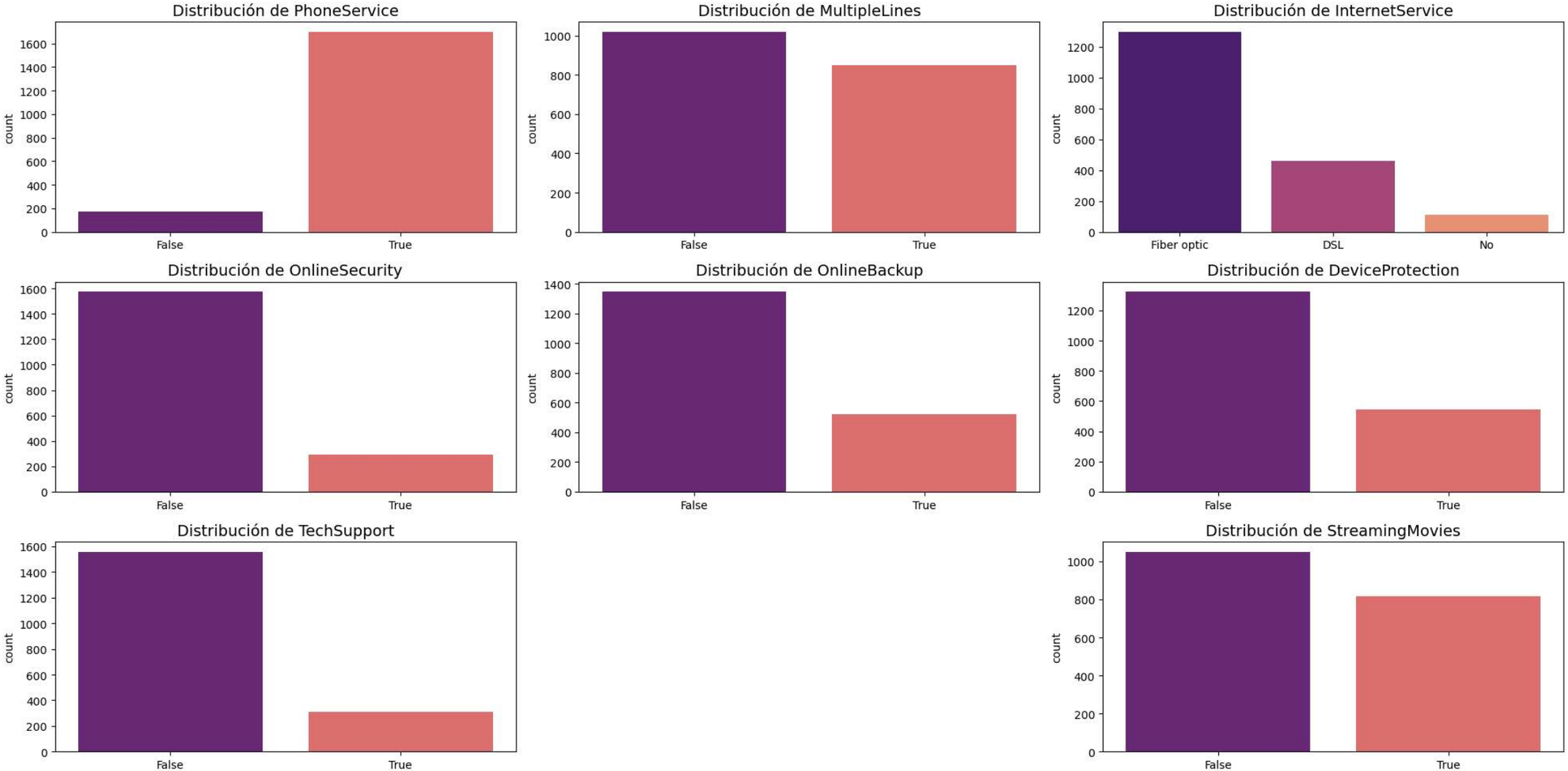
```
plt.tight_layout()
plt.show()
```



In [ ]:
```
tabla_porcentajes = df_merge_clean[['SeniorCitizen', 'Partner', 'Dependents']].apply(lambda x: x.value_counts(normalize=True))
# Multiplicamos por 100 y formateamos para que sea más legible
tabla_porcentajes = (tabla_porcentajes * 100).round(2).T
# Renombramos las columnas para que sean claras
tabla_porcentajes.columns = ['Porcentaje False (%)', 'Porcentaje True (%)']
# Mostramos la tabla
print(tabla_porcentajes)
```

```
              Porcentaje False (%)  Porcentaje True (%)
SeniorCitizen                83.79                16.21
Partner                      51.70                48.30
Dependents                   70.04                29.96
```

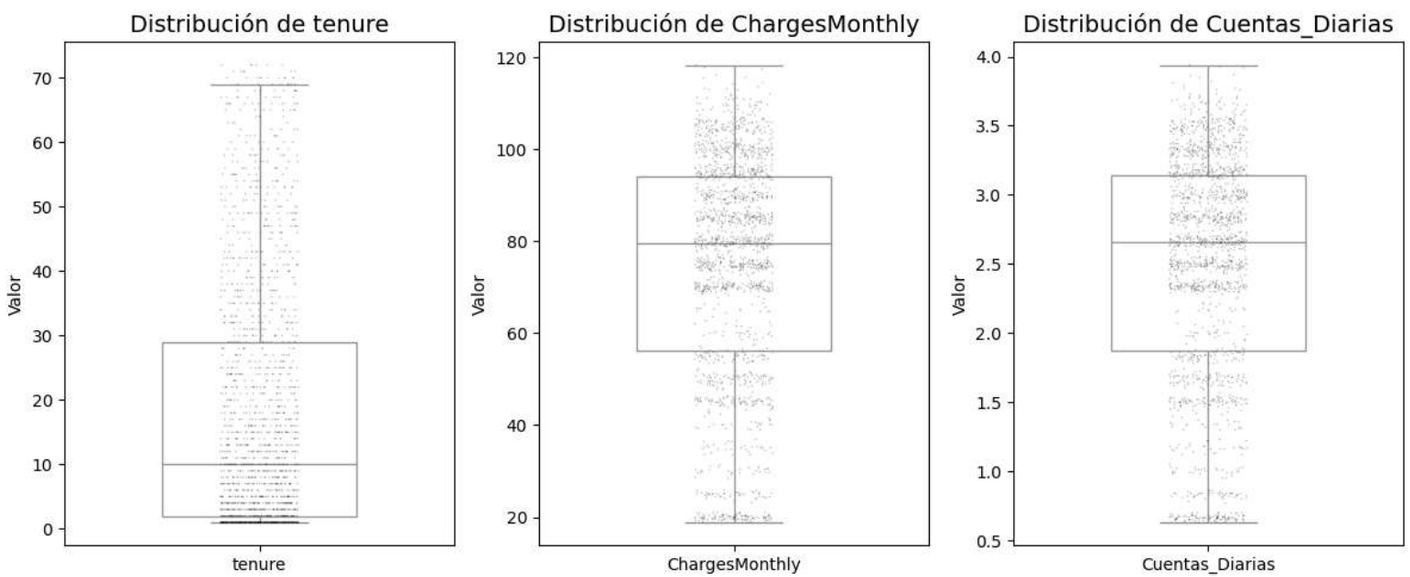
In [42]:
```
analizys_servicios = ['PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']

fig, ax = plt.subplots(3, 3, figsize=(20, 10))
axes = ax.flatten()

# Iteramos para crear cada barplot (count plot)
for i, col in enumerate(analizys_servicios):
    sns.countplot(data=churn_true, x=col, hue=col, ax=axes[i], palette='magma', legend=False)
    axes[i].set_title(f'Distribución de {col}', fontsize=14)
    axes[i].set_xlabel('')

# Eliminamos el último gráfico que queda vacío
fig.delaxes(axes[7])

plt.tight_layout()
plt.show()
```



✅ Explorar cómo las variables numéricas, como "total gastado" o "tiempo de contrato", se distribuyen entre los clientes que cancelaron (evasión) y los que no cancelaron.

In [43]:
```
analizys_boxplot = ['tenure', 'ChargesMonthly', 'Cuentas_Diarias']

fig, ax = plt.subplots(1, 3, figsize=(12, 5))
for i, col in enumerate(analizys_boxplot):
    # Usamos x=[""] * len(churn_true) para crear una base común
    # Y asignamos un nombre fijo al hue para que no se pierda en valores individuales
    sns.boxplot(
        data=churn_true,
```

```
        y=col,
        x=[col] * len(churn_true), # Esto crea una etiqueta única en el eje X
        hue=[col] * len(churn_true), # Esto asigna un color único a esa etiqueta
        ax=ax[i],
        palette='dark:white_r',
        legend=False,
        width=0.5,
        fliersize=0

    )

    # Encima dibujamos el Stripplot (los puntos individuales)
    sns.stripplot(
        data=churn_true,
        y=col,
        ax=ax[i],
        x=[col] * len(churn_true),
        hue=[col] * len(churn_true),
        size=1,        # Puntos muy pequeños
        jitter=True,   # Dispersión horizontal
        alpha=0.3,     # Transparencia
        palette='dark:black',
    )

    ax[i].set_title(f'Distribución de {col}', fontsize=14)
    ax[i].set_xlabel('') # Limpiamos el eje X para que no se vea repetido
    ax[i].set_ylabel('Valor')

plt.tight_layout()
plt.show()
```

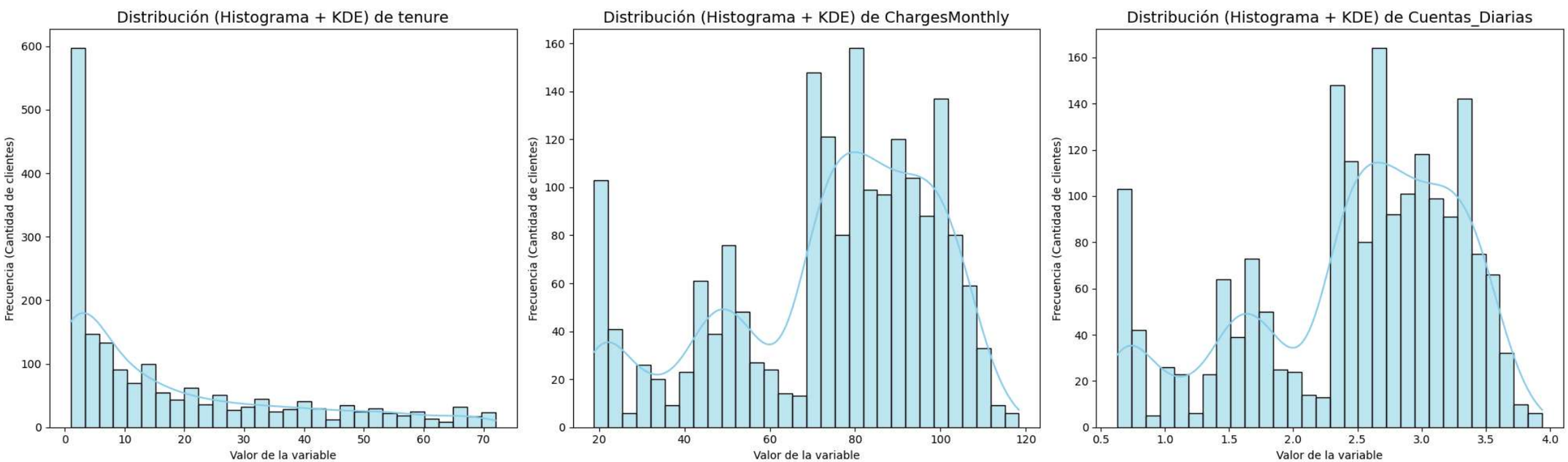


In [44]:
```
churn_true[['tenure','ChargesMonthly','Cuentas_Diarias']].describe()
```

Out[44]:

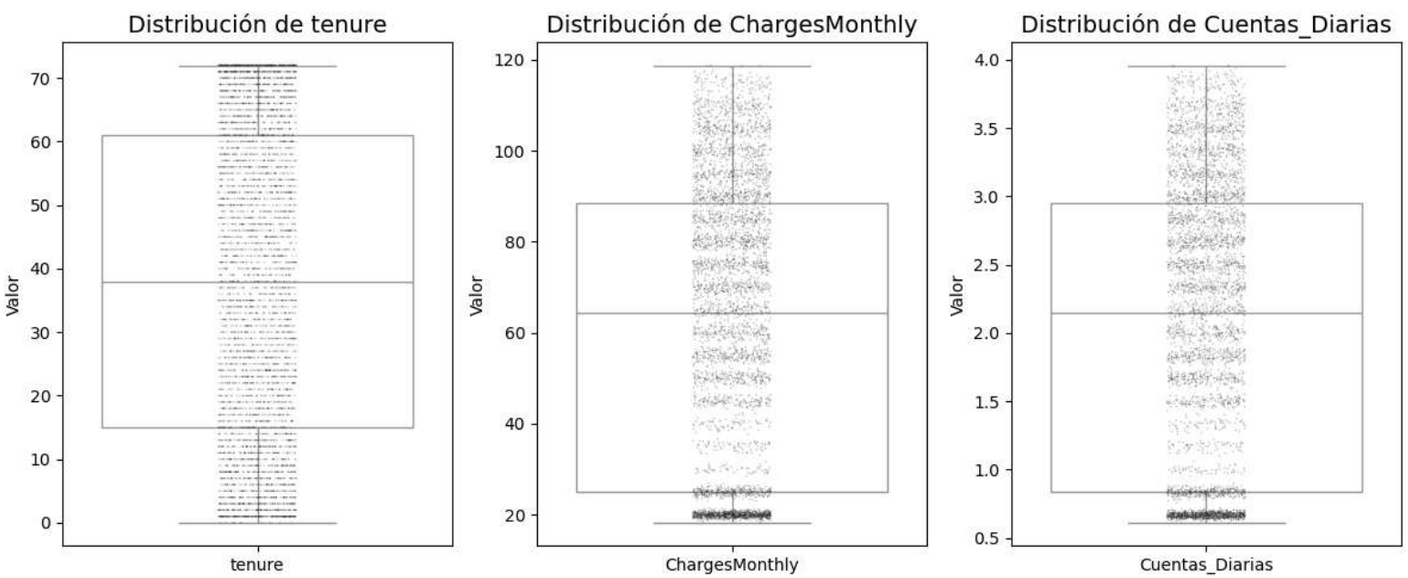
	tenure	ChargesMonthly	Cuentas_Diarias
count	1869.000000	1869.000000	1869.000000
mean	17.979133	74.441332	2.481450
std	19.531123	24.666053	0.822287
min	1.000000	18.850000	0.630000
25%	2.000000	56.150000	1.870000
50%	10.000000	79.650000	2.660000
75%	29.000000	94.200000	3.140000
max	72.000000	118.350000	3.940000

In [45]:
```
fig, axes = plt.subplots(1, 3, figsize=(20, 6))

for i, col in enumerate(analizys_boxplot):
    # Usamos histplot con KDE para ver la "montaña" de densidad
    sns.histplot(data=churn_true, x=col, kde=True, ax=axes[i], color='skyblue', bins=30)

    axes[i].set_title(f'Distribución (Histograma + KDE) de {col}', fontsize=14)
    axes[i].set_ylabel('Frecuencia (Cantidad de clientes)')
    axes[i].set_xlabel('Valor de la variable')

plt.tight_layout()
plt.show()
```



In [46]:
```
analizys_boxplot = ['tenure', 'ChargesMonthly', 'Cuentas_Diarias']
```

```
fig, ax = plt.subplots(1, 3, figsize=(12, 5))
for i, col in enumerate(analizys_boxplot):
    # Usamos x=[""] * len(churn_false) para crear una base común
    # Y asignamos un nombre fijo al hue para que no se pierda en valores individuales
    sns.boxplot(
        data=churn_false,
        y=col,
        x=[col] * len(churn_false), # Esto crea una etiqueta única en el eje X
        hue=[col] * len(churn_false), # Esto asigna un color único a esa etiqueta
        ax=ax[i],
        palette='dark:white_r',
        legend=False
    )

    # Encima dibujamos el Stripplot (los puntos individuales)
    sns.stripplot(
        data=churn_false,
        y=col,
        ax=ax[i],
        x=[col] * len(churn_false),
        hue=[col] * len(churn_false),
        size=1,        # Puntos muy pequeños
        jitter=True,   # Dispersión horizontal
        alpha=0.3,     # Transparencia
        palette='dark:black',
    )

    ax[i].set_title(f'Distribución de {col}', fontsize=14)
    ax[i].set_xlabel('') # Limpiamos el eje X para que no se vea repetido
    ax[i].set_ylabel('Valor')

plt.tight_layout()
plt.show()
```

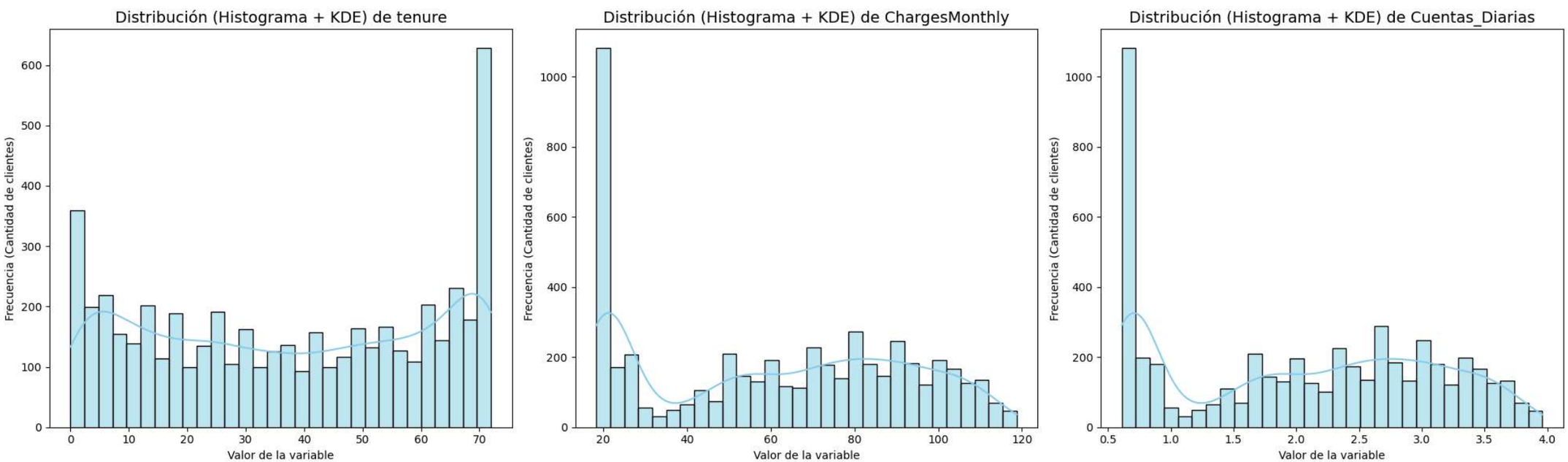


In [47]:
```
churn_false[['tenure','ChargesMonthly','Cuentas_Diarias']].describe()
```

Out[47]:

	tenure	ChargesMonthly	Cuentas_Diarias
count	5174.000000	5174.000000	5174.000000
mean	37.569965	61.265124	2.042080
std	24.113777	31.092648	1.036492
min	0.000000	18.250000	0.610000
25%	15.000000	25.100000	0.840000
50%	38.000000	64.425000	2.150000
75%	61.000000	88.400000	2.950000
max	72.000000	118.750000	3.960000

In [48]:
```
fig, axes = plt.subplots(1, 3, figsize=(20, 6))

for i, col in enumerate(analizys_boxplot):
    # Usamos histplot con KDE para ver la "montaña" de densidad
    sns.histplot(data=churn_false, x=col, kde=True, ax=axes[i], color='skyblue', bins=30)

    axes[i].set_title(f'Distribución (Histograma + KDE) de {col}', fontsize=14)
    axes[i].set_ylabel('Frecuencia (Cantidad de clientes)')
    axes[i].set_xlabel('Valor de la variable')

plt.tight_layout()
plt.show()
```



📊 Análisis de correlación entre variables

> Explorar la correlación entre diferentes variables del dataset. Esto puede ayudar a identificar qué factores tienen mayor relación con la evasión de clientes, > como:
>
> - La relación entre la cuenta diaria y la evasión.
> - Cómo la cantidad de servicios contratados afecta la probabilidad de churn.

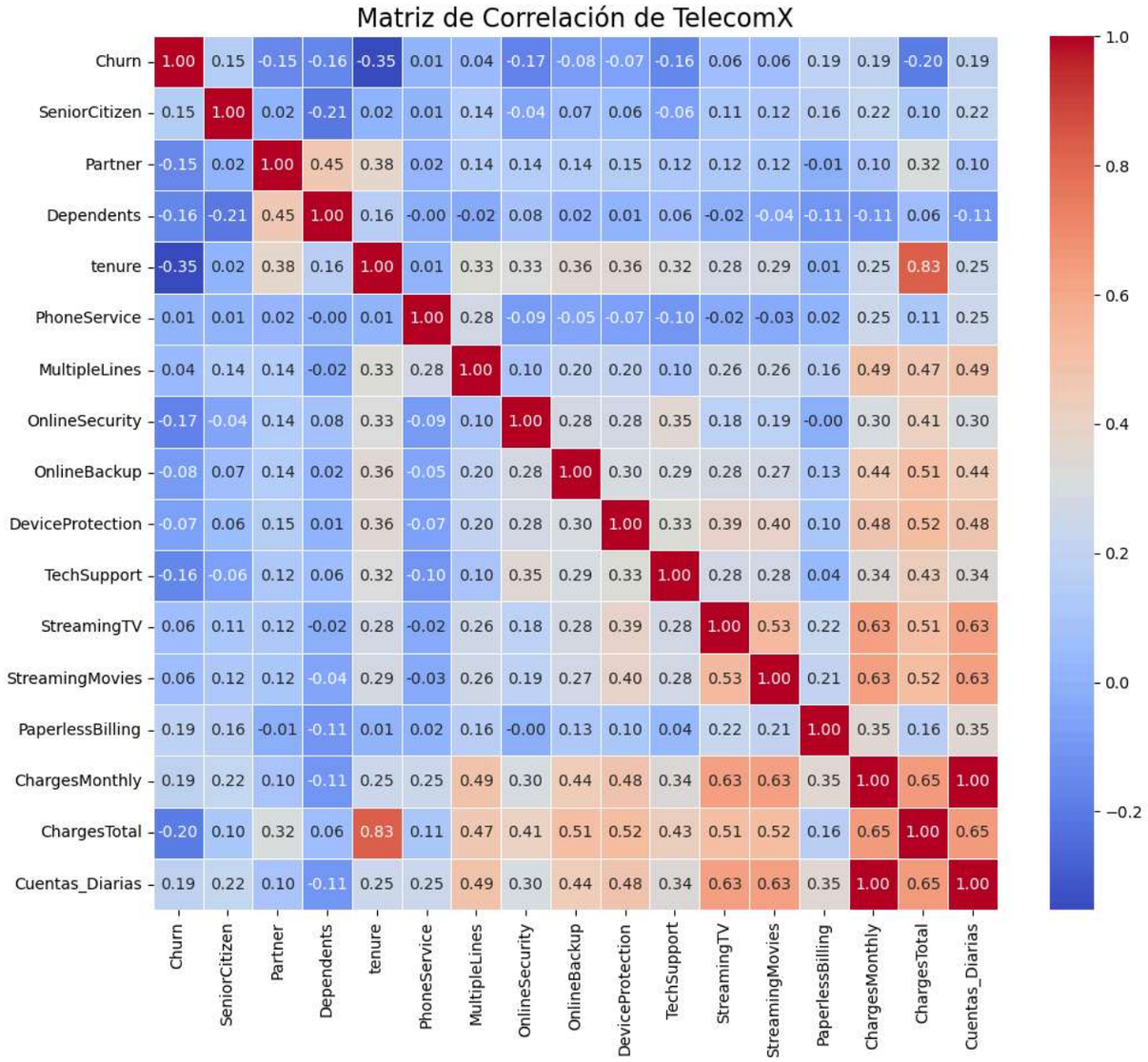
In [49]:
```
# Seleccionamos solo las columnas numéricas y booleanas
corr_matrix = df_merge_clean.select_dtypes(include=['number', 'bool']).corr()
corr_matrix
```

Out[49]:

	Churn	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	PaperlessBilling	ChargesMonthly	ChargesTotal	Cuentas_Diarias
Churn	1.000000	0.150889	-0.150448	-0.164221	-0.352229	0.011942	0.040102	-0.171226	-0.082255	-0.066160	-0.164674	0.063228	0.061382	0.191825	0.193356	-0.198324	0.193412
SeniorCitizen	0.150889	1.000000	0.016479	-0.211185	0.016567	0.008576	0.142948	-0.038653	0.066572	0.059428	-0.060625	0.105378	0.120176	0.156530	0.220173	0.103006	0.220147
Partner	-0.150448	0.016479	1.000000	0.452676	0.379697	0.017706	0.142057	0.143106	0.141498	0.153786	0.119999	0.124666	0.117412	-0.014877	0.096848	0.317504	0.096909
Dependents	-0.164221	-0.211185	0.452676	1.000000	0.159712	-0.001762	-0.024526	0.080972	0.023671	0.013963	0.063268	-0.016558	-0.039741	-0.111377	-0.113890	0.062078	-0.113939
tenure	-0.352229	0.016567	0.379697	0.159712	1.000000	0.008448	0.331941	0.327203	0.360277	0.360653	0.324221	0.279756	0.286111	0.006152	0.247900	0.826178	0.247910
PhoneService	0.011942	0.008576	0.017706	-0.001762	0.008448	1.000000	0.279690	-0.092893	-0.052312	-0.071227	-0.096340	-0.022574	-0.032959	0.016505	0.247398	0.113214	0.247361
MultipleLines	0.040102	0.142948	0.142057	-0.024526	0.331941	0.279690	1.000000	0.098108	0.202237	0.201137	0.100571	0.257152	0.258751	0.163530	0.490434	0.468504	0.490457
OnlineSecurity	-0.171226	-0.038653	0.143106	0.080972	0.327203	-0.092893	0.098108	1.000000	0.283832	0.275438	0.354931	0.176207	0.187398	-0.003636	0.296594	0.411651	0.296591
OnlineBackup	-0.082255	0.066572	0.141498	0.023671	0.360277	-0.052312	0.202237	0.283832	1.000000	0.303546	0.294233	0.282106	0.274501	0.126735	0.441780	0.509226	0.441762
DeviceProtection	-0.066160	0.059428	0.153786	0.013963	0.360653	-0.071227	0.201137	0.275438	0.303546	1.000000	0.333313	0.390874	0.402111	0.103797	0.482692	0.521983	0.482648
TechSupport	-0.164674	-0.060625	0.119999	0.063268	0.324221	-0.096340	0.100571	0.354931	0.294233	0.333313	1.000000	0.278070	0.279358	0.037880	0.338304	0.431883	0.338300
StreamingTV	0.063228	0.105378	0.124666	-0.016558	0.279756	-0.022574	0.257152	0.176207	0.282106	0.390874	0.278070	1.000000	0.533094	0.223841	0.629603	0.514973	0.629604
StreamingMovies	0.061382	0.120176	0.117412	-0.039741	0.286111	-0.032959	0.258751	0.187398	0.274501	0.402111	0.279358	0.533094	1.000000	0.211716	0.627429	0.520122	0.627402
PaperlessBilling	0.191825	0.156530	-0.014877	-0.111377	0.006152	0.016505	0.163530	-0.003636	0.126735	0.103797	0.037880	0.223841	0.211716	1.000000	0.352150	0.158574	0.352135
ChargesMonthly	0.193356	0.220173	0.096848	-0.113890	0.247900	0.247398	0.490434	0.296594	0.441780	0.482692	0.338304	0.629603	0.627429	0.352150	1.000000	0.651174	0.999996
ChargesTotal	-0.198324	0.103006	0.317504	0.062078	0.826178	0.113214	0.468504	0.411651	0.509226	0.521983	0.431883	0.514973	0.520122	0.158574	0.651174	1.000000	0.651189
Cuentas_Diarias	0.193412	0.220147	0.096909	-0.113939	0.247910	0.247361	0.490457	0.296591	0.441762	0.482648	0.338300	0.629604	0.627402	0.352135	0.999996	0.651189	1.000000

📊 Matriz de Correlacion

In [50]:
```
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

plt.title('Matriz de Correlación de TelecomX', fontsize=16)
plt.show()
```



📊 Correlacion directa con la evacion.

In [51]:
```
# Ordenamos de mayor a menor para que el gráfico sea legible y dejaamos solo la correlacion con Churn.
churn_corr = corr_matrix['Churn'].sort_values(ascending=False).drop('Churn').reset_index()
churn_corr.sample(5)
```

Out[51]:

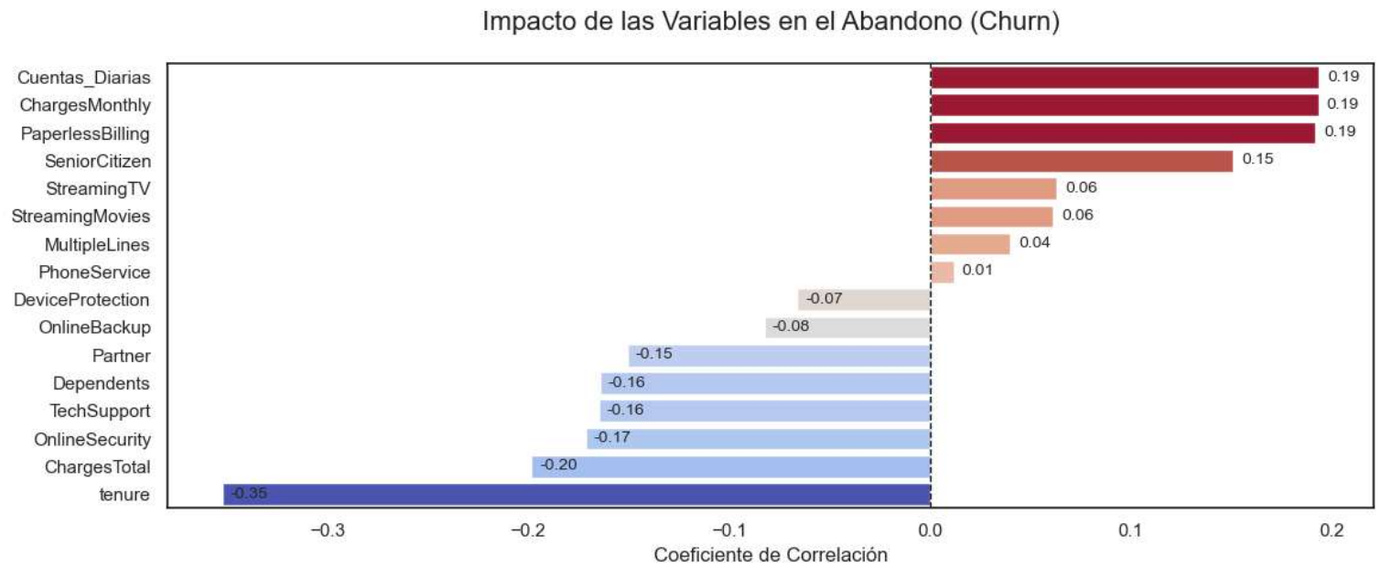
	index	Churn
6	MultipleLines	0.040102
5	StreamingMovies	0.061382
8	DeviceProtection	-0.066160
10	Partner	-0.150448
14	ChargesTotal	-0.198324

In [52]:
```
# Renombramos las columnas
churn_corr.columns = ['Variable', 'Correlacion']
churn_corr.sample(5)
```

Out[52]:

	Variable	Correlacion
11	Dependents	-0.164221
9	OnlineBackup	-0.082255
15	tenure	-0.352229
6	MultipleLines	0.040102
13	OnlineSecurity	-0.171226

In [53]:
```
sns.set_theme(style='white')

plt.figure(figsize=(12, 5))

ax = sns.barplot(
    data=churn_corr,
    x='Correlacion',
    y='Variable',
    hue='Correlacion',
    palette='coolwarm',
    legend=False
)

plt.title('Impacto de las Variables en el Abandono (Churn)', fontsize=16, pad=20)
plt.xlabel('Coeficiente de Correlación', fontsize=12)
plt.ylabel(' ')

# Añadimos una línea vertical en 0 para separar claramente los bandos
plt.axvline(0, color='black', lw=1, ls='--')

# Añadimos el valor numérico al final de cada barra
for i, p in enumerate(ax.patches):
    ax.annotate(f'{p.get_width():.2f}', (p.get_width(), p.get_y() + p.get_height()/2),
                ha='left', va='center', xytext=(5, 0),
                textcoords='offset points', fontsize=10)

plt.tight_layout()
plt.show()
```



Informe final

Informe Técnico: Análisis de Evasión de Clientes (Churn) - TelecomX

1. Introducción.

El objetivo de este análisis es identificar los factores críticos que influyen en la pérdida de clientes **(Chum)** de la compañía ***TelecomX***. La evasión de clientes es un desafío estratégico; retener a un usuario existente es significativamente más rentable que adquirir uno nuevo. A través de este estudio, buscamos patrones en el comportamiento de facturación, servicios contratados y antigüedad para proponer estrategias de retención basadas en datos.

2. Limpieza y Tratamiento de Datos

Para garantizar la integridad del análisis, se realizaron los siguientes pasos de pre-procesamiento:

- Importación Estable: Se utilizó la librería requests para descargar el dataset desde un repositorio remoto de GitHub, evitando errores de conexión incompleta (IncompleteRead).

```
response = requests.get(url)
content = response.content
df = pd.read_json(io.StringIO(content.decode('utf-8')))
df
```

- Normalización JSON: Se normaliza el documento .json con el fin de generar una base de datos tratable para realizar el analisis de acuerdo a lo solicitado.

```
df_merge = columnas_01
for i in columnas_02:
    column = pd.json_normalize(columnas_02[i])
    df_merge = pd.concat([df_merge, column], axis=1)
df_merge.info()
```

- Conversión de Tipos:
  - La columna Charges.Total fue convertida a float, gestionando valores vacíos que impedían el cálculo.

- Se transformaron 12 columnas categóricas (Yes/No) a tipo Booleano (True/False) para optimizar el uso de memoria y facilitar el análisis estadístico.

- Gestión de Valores Faltantes: Se identificaron y eliminaron filas con datos nulos en la columna objetivo (Churn) mediante el método .dropna() con .copy() para evitar advertencias de fragmentación de memoria.
  - En este punto se genera una base de datos nueva y limpia para realizar el Analisis exploratorio de datos.

```
df_merge_clean = df_merge[df_merge['Churn'] != ''].copy()
df_merge_clean.sample(5)
```

- Ingeniería de Características: Se creó la métrica Cuentas_Diarias, calculada como Charges.Monthly / 30, para analizar la facturación a un nivel de granularidad más detallado.

```
df_merge_clean['Cuentas_Diarias'] = (df_merge_clean['ChargesMonthly']/30).round(2)
df_merge_clean
```

3. Análisis Exploratorio de Datos (EDA)

📊 Distribución y Densidad

- El 26.53% de los usuarios ya no cuentan los nuestros servicios de acuerdo a la informacion generada, `(df_merge_clean[df_merge_clean['Churn'] == True])`
- La evacion generada distribuida segun categorias como Genero, Tipo de contrato y metodo de pago, `(evacion = ['gender','Contract','PaymentMethod'])`, muestra que:
  - El genero del evasor no tiene ningun tipo de peso.

```
gender
Male      50.4756
Female    49.5244
```

  - Los usuarios que mas se retiran de los servicios son quienes el tipo de contratacion es pago mes a mes.

```
Contract
Month-to-month    55.0192
Two year          24.0664
One year          20.9144
```

  - El tipo de pago mas usado por quienes se retiran del servicio es, Validacion Electronica.

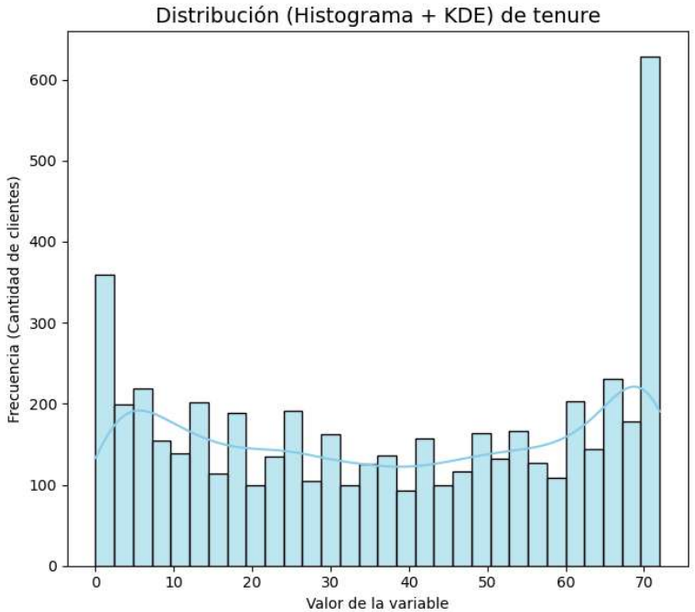
```
PaymentMethod
Electronic check             33.5794
Mailed check                 22.8880
Bank transfer (automatic)    21.9225
Credit card (automatic)      21.6101
```

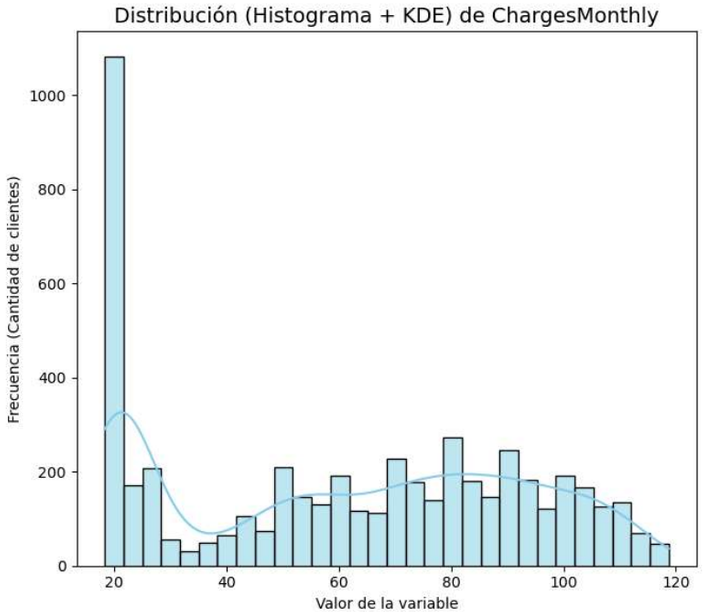
- El analisis realizado sobre datos familiares personales como si el ciente es un cliente de la tercera de edad, si se encuentra o no casado o si tiene o no personas que dependen de el, `['SeniorCitizen', 'Partner', 'Dependents']`, muestran que estos datos no son relevantes, pues en su mayoria la respuesta fue negativa.

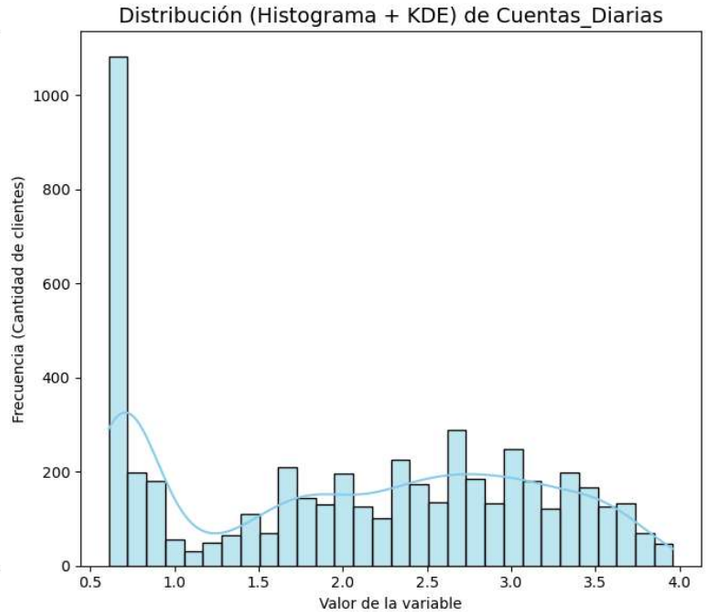
```
               Porcentaje False (%)  Porcentaje True (%)
SeniorCitizen                 83.79                16.21
Partner                       51.70                48.30
Dependents                    70.04                29.96
```

Utilizamos Histogramas con estimación de densidad de kernel (KDE) para entender la distribución de los clientes que no se retiran:

- Se detectó una acumulación masiva de clientes en el rango de los 20 USD (Cargos Mensuales), lo que indica un segmento dominante de usuarios con planes básicos.



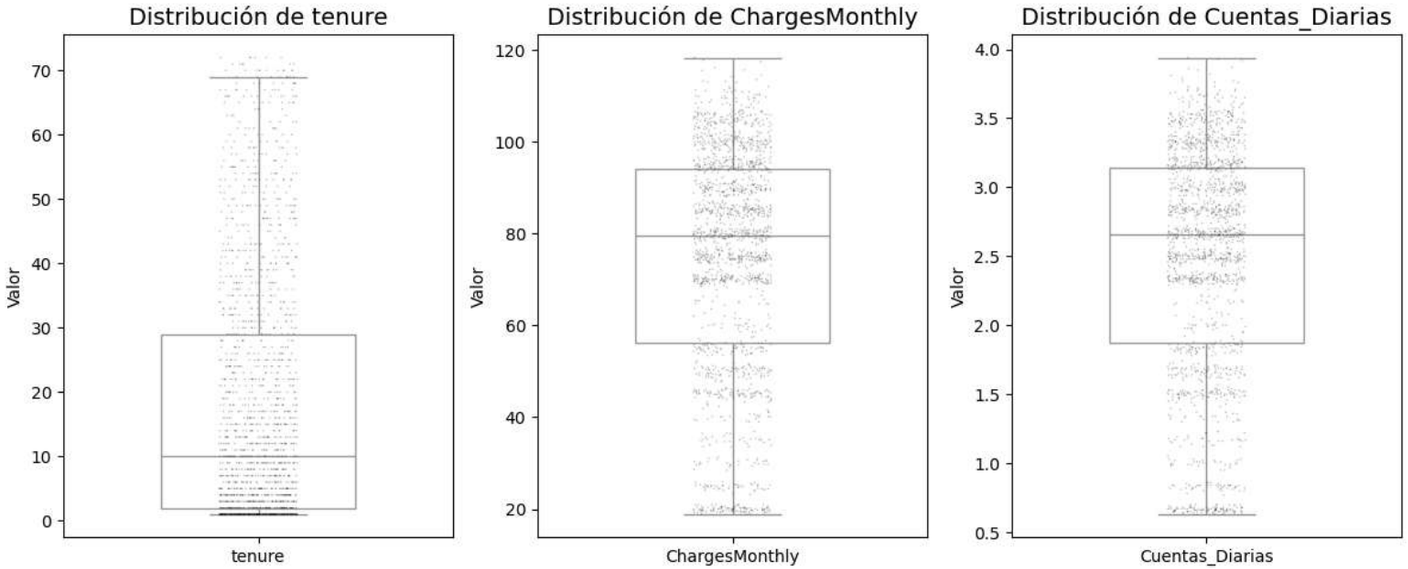




- Antigüedad (Tenure): La distribución es bimodal, concentrándose en clientes muy nuevos y clientes de larga trayectoria.

Y validamos esta informacion comparandola con los clientes que se retiran y observamos:

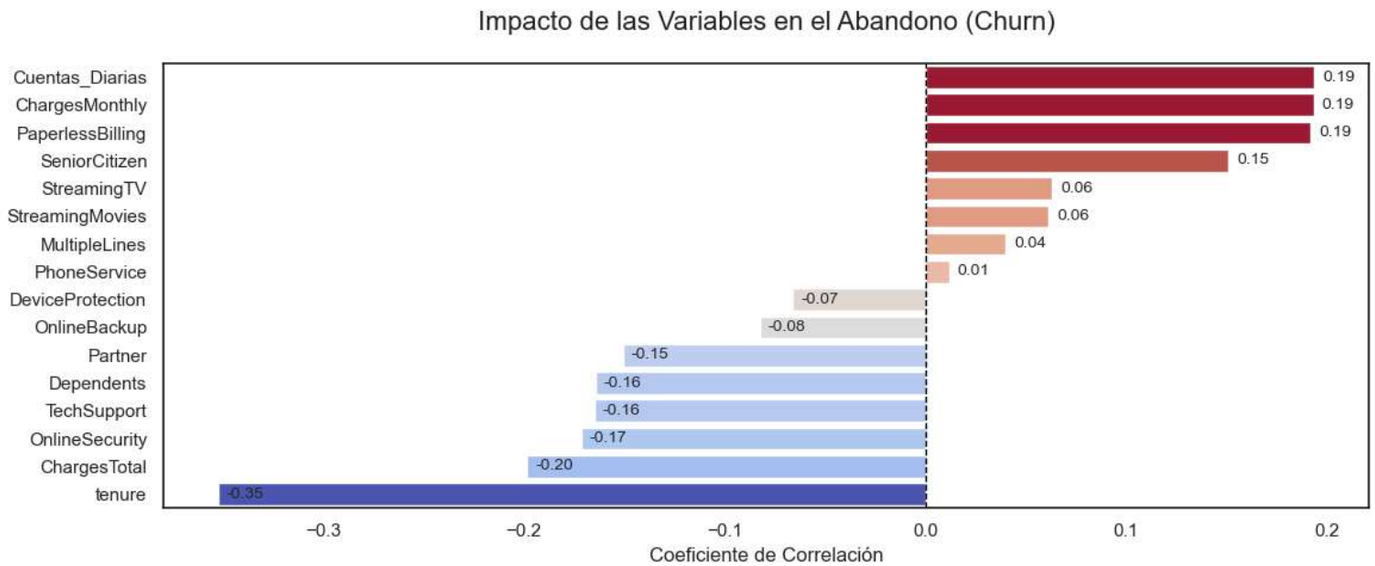
- Los clientes que se retrian son clientes que en su mayoria no duran mas de 30 meses de contrato.



- El valor que se paga mensual de los clientes que se retiran tiene una media de 80 USD.

Análisis de Correlación

Implementamos una Matriz de Correlación con Seaborn (utilizando la paleta coolwarm) para medir el impacto de cada variable en el Churn:



- Correlación Positiva: Los cargos mensuales elevados están directamente relacionados con una mayor probabilidad de abandono.
- Correlación Negativa: La antigüedad (tenure) es el principal factor de retención; a mayor tiempo con la empresa, menor es el riesgo de evasión.

4. Conclusiones e Insights

- Sensibilidad al Precio: Los clientes con cargos mensuales altos presentan un riesgo de fuga significativamente mayor. El pico de usuarios en los 20 USD sugiere que la estabilidad del negocio reside en los planes de entrada.
- El Factor Lealtad: Los primeros meses de contrato son críticos. Una vez que el cliente supera la barrera inicial de antigüedad, la probabilidad de Churn cae drásticamente.
- Métricas Diarias: La nueva métrica Cuentas_Diarias permitió validar que no existen anomalías o outliers extremos que distorsionen los promedios de cobro.

5. Recomendaciones Estratégicas

1. Programas de Fidelización Temprana: Implementar incentivos o descuentos durante los primeros 6 meses de contrato para reducir el Churn en clientes nuevos.
2. Alertas por Cargos Elevados: Monitorear a los clientes cuyos cargos mensuales superen el promedio del segmento (aprox. 65 USD) y ofrecerles planes empaquetados más competitivos.
3. Migración de Métodos de Pago: Fomentar el uso de tarjetas de crédito o transferencias automáticas, ya que suelen estar asociados a una menor tasa de olvido o deserción.