1. Preprocesamiento de Datos

In [1]:
```
#Importamos librerias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


# Cargamos datasets
calls_df = pd.read_csv('/datasets/telecom_dataset_us.csv')
clients_df = pd.read_csv('/datasets/telecom_clients_us.csv')
```

In [2]:
```
# Mostramos los datos de los datasets al igual que verificamos los tipos de datos y
datasets = {
    'calls_df': calls_df,
    'clients_df': clients_df}

# Verificamos los datos
for name, df in datasets.items():
    print(f"\n{name} info:")
    print(df.info())
    print(f"\nValores nulos en {name}:")
    print(df.isnull().sum())
    print(f"\nDuplicados en {name}:")
    print(df.duplicated().sum())
```

```
calls_df info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53902 entries, 0 to 53901
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   user_id              53902 non-null  int64  
 1   date                 53902 non-null  object 
 2   direction            53902 non-null  object 
 3   internal             53785 non-null  object 
 4   operator_id          45730 non-null  float64
 5   is_missed_call       53902 non-null  bool   
 6   calls_count          53902 non-null  int64  
 7   call_duration        53902 non-null  int64  
 8   total_call_duration  53902 non-null  int64  
dtypes: bool(1), float64(1), int64(4), object(3)
memory usage: 3.3+ MB
None

Valores nulos en calls_df:
user_id                   0
date                      0
direction                 0
internal                117
operator_id            8172
is_missed_call            0
calls_count               0
call_duration             0
total_call_duration       0
dtype: int64

Duplicados en calls_df:
4900

clients_df info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 732 entries, 0 to 731
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   user_id      732 non-null    int64 
 1   tariff_plan  732 non-null    object
 2   date_start   732 non-null    object
dtypes: int64(1), object(2)
memory usage: 17.3+ KB
None

Valores nulos en clients_df:
user_id        0
tariff_plan    0
date_start     0
dtype: int64

Duplicados en clients_df:
0
```

In [3]:
```
# Visualizamos los datasets
display(calls_df.sample(10))
display(clients_df.sample(10))
```

	user_id	date	direction	internal	operator_id	is_missed_call	calls_count	ca
2594	166407	2019-11-11 00:00:00+03:00	out	True	888532.0	False	7	
53883	168603	2019-11-20 00:00:00+03:00	out	False	959118.0	True	3	
15227	166896	2019-09-27 00:00:00+03:00	in	False	895576.0	False	1	
38899	167654	2019-11-10 00:00:00+03:00	out	False	918978.0	True	4	
41542	167888	2019-10-11 00:00:00+03:00	out	False	928888.0	False	5	
44441	168047	2019-10-30 00:00:00+03:00	out	False	937604.0	False	1	
34844	167521	2019-10-05 00:00:00+03:00	out	False	919794.0	False	1	
32287	167456	2019-11-19 00:00:00+03:00	out	False	921102.0	False	1	
28098	167176	2019-11-14 00:00:00+03:00	out	True	907970.0	False	3	
16307	166916	2019-10-07 00:00:00+03:00	in	False	906410.0	False	7	

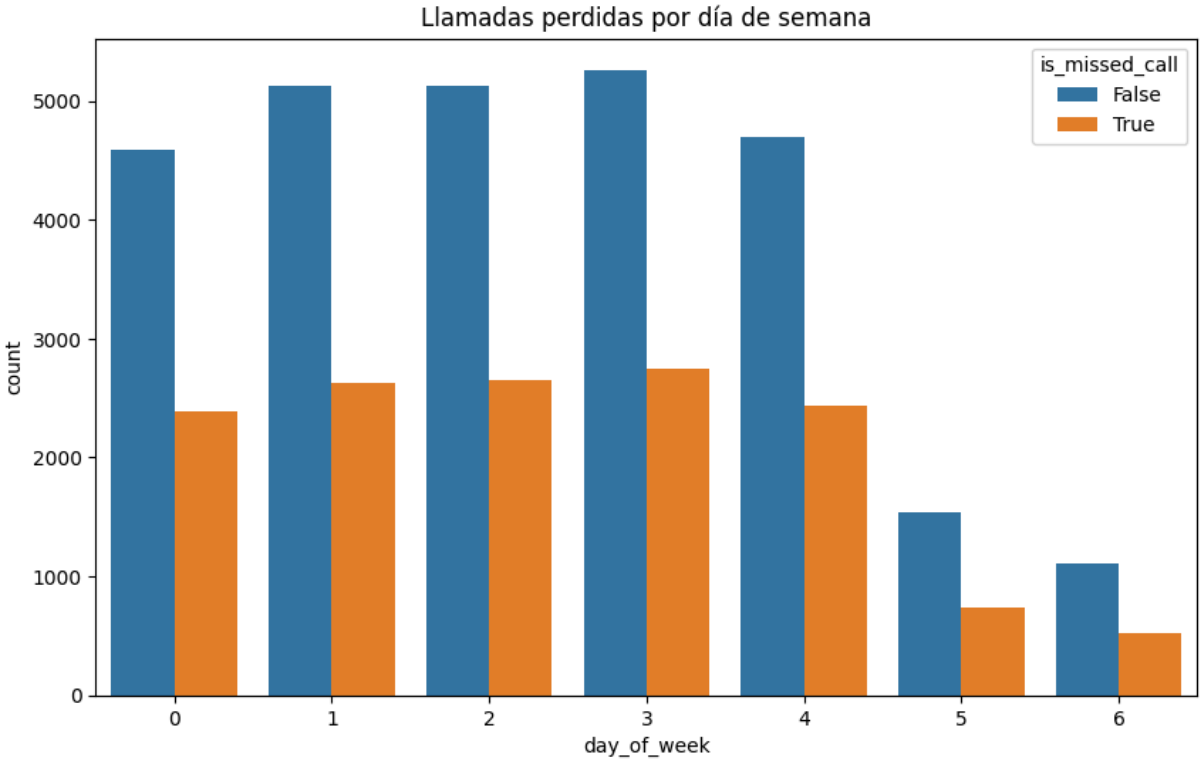
	user_id	tariff_plan	date_start
205	167501	C	2019-09-18
675	168578	B	2019-10-31
161	168206	C	2019-10-16
436	166505	C	2019-08-06
226	167936	C	2019-10-07
363	166717	C	2019-08-16
412	167112	C	2019-09-02
477	167652	B	2019-09-24
270	167618	C	2019-09-23
507	166495	B	2019-08-06

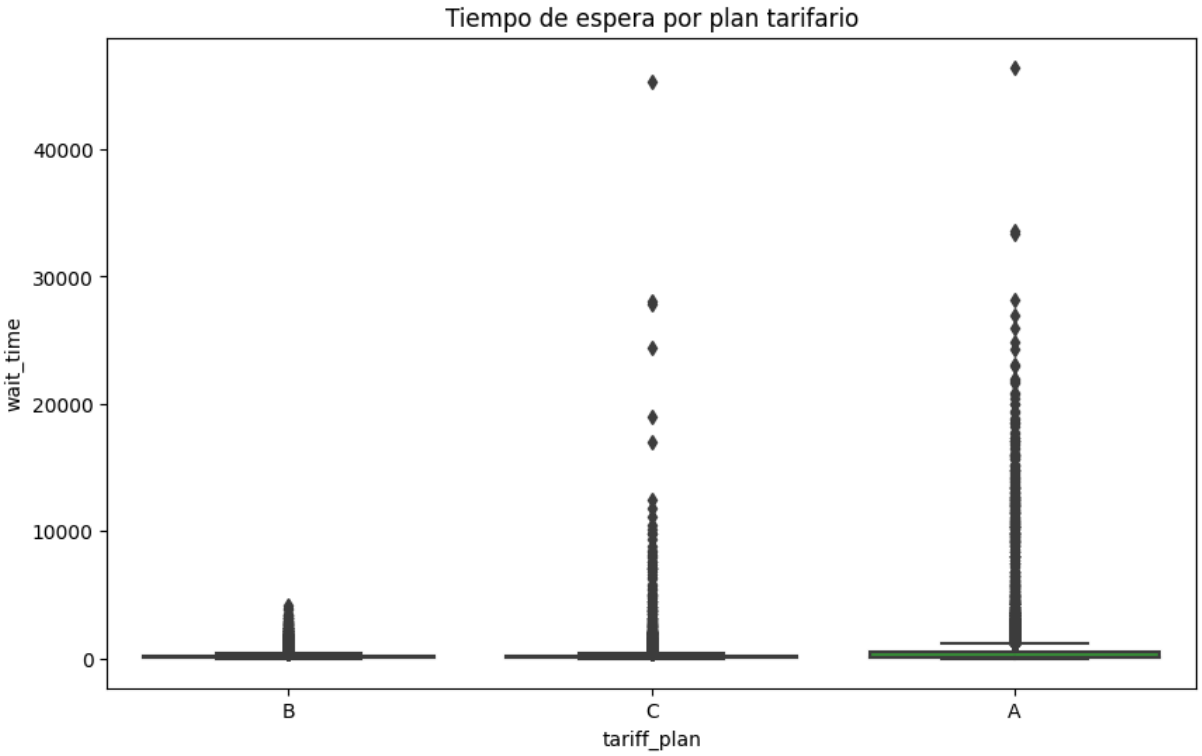
In [4]:
```
# Unimos los data sets por user_id
data = pd.merge(calls_df, clients_df, on='user_id', how='left')

# Convertimos las fechas a datetime
data['date'] = pd.to_datetime(data['date'])
data['date_start'] = pd.to_datetime(data['date_start'])
# Cambiamos los valores nulos
data['internal'] = data['internal'].fillna('Unknown')  # Asignamos como categoría d
#data['operator_id'] = data['operator_id'].fillna(0)  # Asignamos 0 para operadores

data = data.dropna(subset=['operator_id']).copy()

data = data.drop_duplicates()
```

Se encontraron 4,900 registros duplicados en el dataset de llamadas (calls_df), ademas que se encontraron valores nulos significativos en operator_id (8,172) y internal (117), el dataset de clientes (clients_df) estaba completo sin duplicados. Se realizó una fusión de datasets y tratamiento de valores nulos, especificamente en operator_id (8,172) y internal (117), asignando categoría desconocida a internal y adignando 0 para operadores desconocidos.

2. Análisis Exploratorio de Datos (EDA)

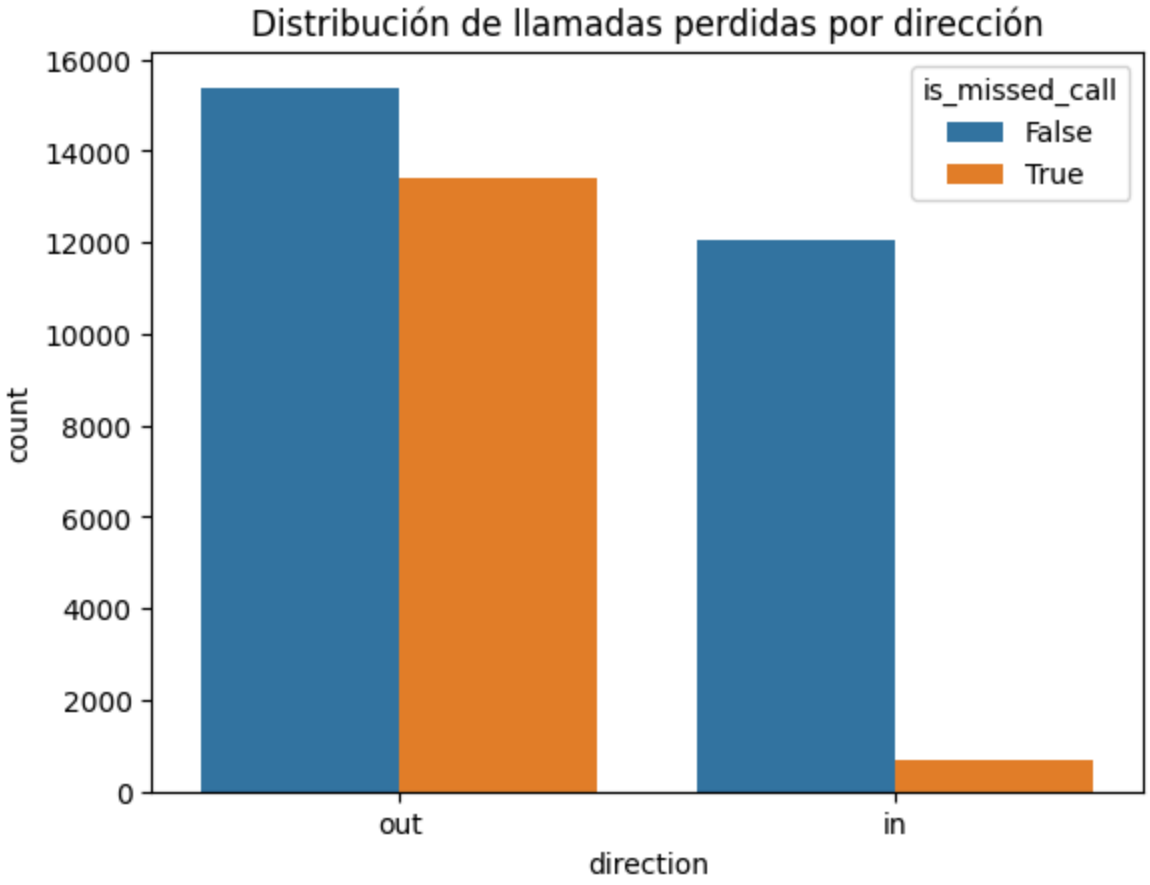
In [5]:
```
# Filtramos para ver por hora y dia de la semana
data['day_of_week'] = data['date'].dt.dayofweek
data['hour'] = data['date'].dt.hour

# Gráficamos las llamadas por día de semana
plt.figure(figsize=(10, 6))
sns.countplot(data=data, x='day_of_week', hue='is_missed_call')
plt.title('Llamadas perdidas por día de semana')
plt.show()

# Calculamos el tiempo de espera
data['wait_time'] = data['total_call_duration'] - data['call_duration']

# Graficamos para ver la relación con plan tarifario
plt.figure(figsize=(10, 6))
sns.boxplot(data=data, x='tariff_plan', y='wait_time')
plt.title('Tiempo de espera por plan tarifario')
plt.show()
```

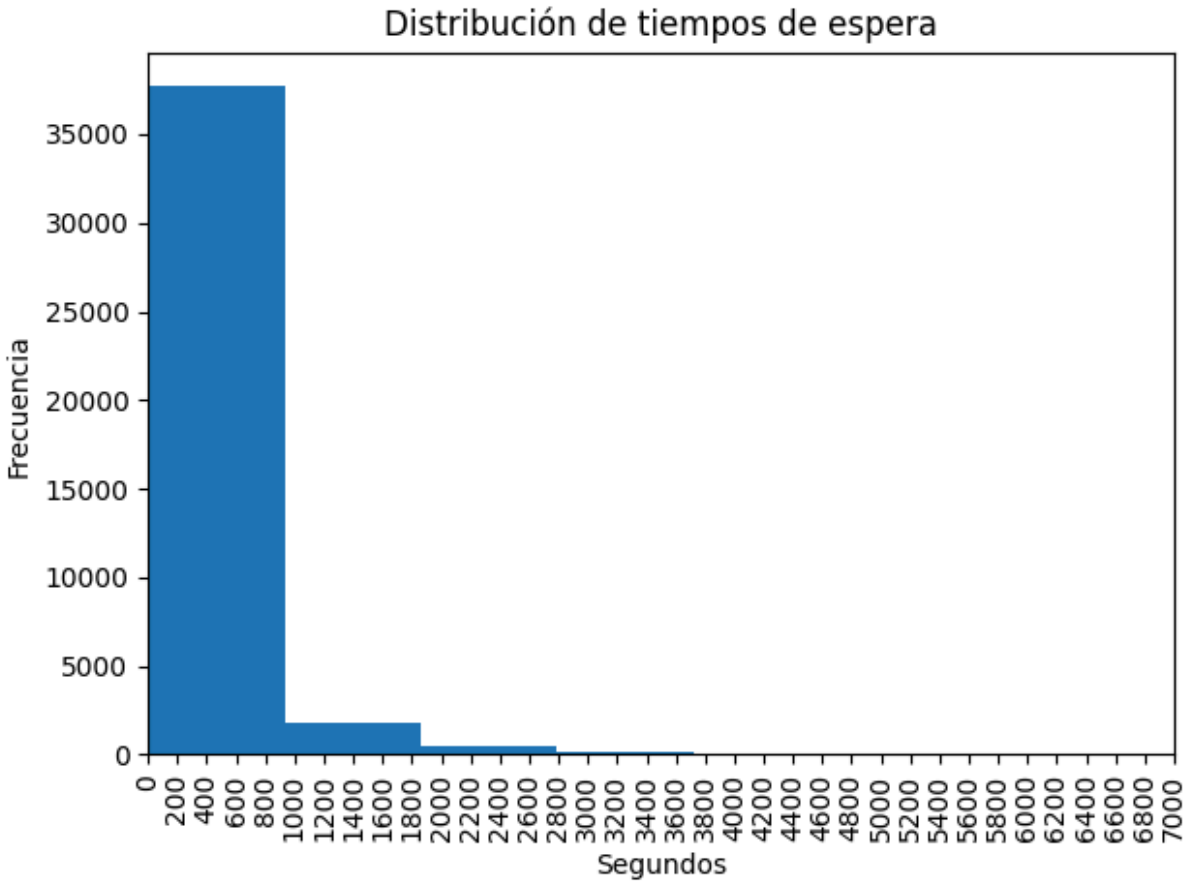




Pudimos encontrar que tenemos un mayor volumen de llamadas los Día 1 y (Martes y Jueves) con un numero mayor a 5000 llamadas pico semanal.Mientras que el dia 5 y 6 (Sábado y Domingo) con cerca de 1000 llamadas, habiendo una mayor proporción en llamadas entrantes (in) que salientes (out). Ademas que pudimos encontrar que en el tiempo de espera por plan tarifario, podemos encontrar diferencias notables entre planes, siendo el plan "a" el que mas tiempo de espera suele alcanzar.

In [6]:
```
# Calculamos el tiempo de espera
data['wait_time'] = data['total_call_duration'] - data['call_duration']

# Graficamos para ver las llamadas perdidas por dirección (in y out)
sns.countplot(data=data, x='direction', hue='is_missed_call')
plt.title('Distribución de llamadas perdidas por dirección')
plt.show()

# Histograma de tiempos de espera
plt.hist(data[data['wait_time'] > 0]['wait_time'], bins=50)
plt.xlim(0, 7000)
plt.xticks(range(0, 7001, 200))
plt.xticks(rotation=90)
plt.title('Distribución de tiempos de espera')
plt.xlabel('Segundos')
plt.ylabel('Frecuencia')
plt.tight_layout()
plt.show()
```





Podemos ver una una mayor proporción en llamadas perdidas en llamadas entrantes (in) que salientes (out).Junto con que los tiempos de espera tienen una mayor frecuencia entre las 15 minutos (900 segundos), lo que es preocupante,ya que tienen una gran cantidad de tiempo.

Análisis por operador:

In [7]:
```
# Creamos la variable de experiencia del operador (días desde primera llamada)
data['operator_experience'] = (data['date'] - data.groupby('operator_id')['date'].t

# Calculamos las métricas por operador
operator_stats = data.groupby('operator_id').agg({
    'is_missed_call': 'mean',
    'wait_time': 'mean',
    'calls_count': 'sum',
    'operator_experience': 'max',
    'direction': lambda x: (x == 'out').sum()
}).reset_index()

# Calculamos las metricas por operador
operator_stats.columns = [
    'operator_id',          # Columna del groupby
    'tasa_perdidas',        # is_missed_call
    'tiempo_espera_promedio',  # wait_time
    'total_llamadas',       # calls_count
    'experiencia_operador', # operator_experience
```

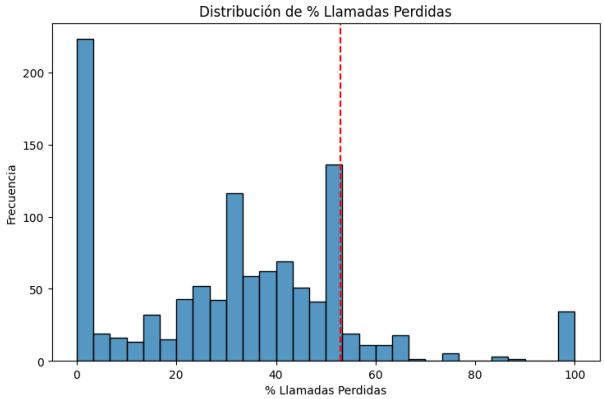
```
    'llamadas_salientes' # direction == 'out'
]

# Calculamos el porcentaje de llamadas perdidas.
operator_stats['porcentaje_perdidas'] = operator_stats['tasa_perdidas'] * 100
```

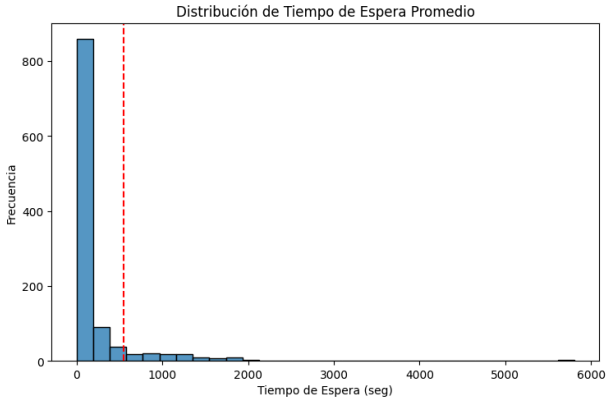
Identificación de operadores ineficaces

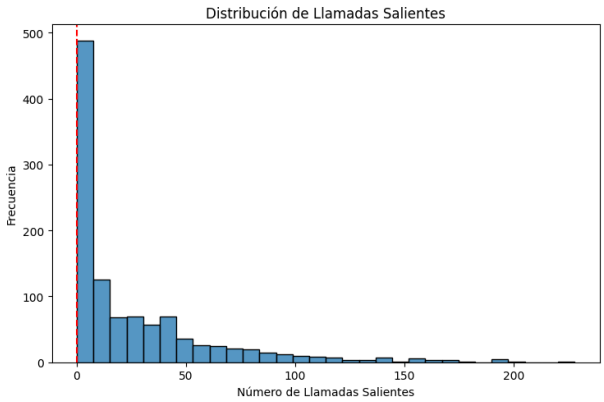
Tomamos percentil_perdidas=90 refiriendonos a un alto porcentaje de llamadas perdidas significando algo malo, percentil_espera=90 refiriendonos a que el alto tiempo de espera significa algo malo y finalmente tomamos el percentil_salientes=10 refiriendonos a que un bajo número de llamadas hechas significa malo.

In [8]:
```
# Hacemos la función para identificar a los operadores ineficaces
def identificar_operadores_ineficaces(metricas_operador, percentil_perdidas=90,
                                      percentil_espera=90, percentil_salientes=10):
    # Creamos el DataFrame para resultados
    resultados = metricas_operador.copy()

    # Calculamos los umbrales
    umbral_perdidas = np.percentile(resultados['porcentaje_perdidas'], percentil_pe
    umbral_espera = np.percentile(resultados['tiempo_espera_promedio'], percentil_e
    umbral_salientes = np.percentile(resultados['llamadas_salientes'], percentil_sa

    # Identificamos a los operadores problemáticos en cada métrica
    resultados['alto_perdidas'] = resultados['porcentaje_perdidas'] >= umbral_perdi
    resultados['alta_espera'] = resultados['tiempo_espera_promedio'] >= umbral_espe
    resultados['bajas_salientes'] = resultados['llamadas_salientes'] <= umbral_sali

    # Calculamos la puntuación de ineficacia (cuántas métricas problemáticas tiene)
    resultados['puntuacion_ineficacia'] = (resultados['alto_perdidas'].astype(int)
                                           resultados['alta_espera'].astype(int) +
                                           resultados['bajas_salientes'].astype(int))

    # Clasificamos como ineficaz si tiene al menos 2 métricas problemáticas
    resultados['ineficaz'] = resultados['puntuacion_ineficacia'] >= 2

    return resultados, (umbral_perdidas, umbral_espera, umbral_salientes)

##  Visualizamos los resultados
def visualizar_resultados(metricas_operador, resultados, umbrales):
    umbral_perdidas, umbral_espera, umbral_salientes = umbrales

    # Configurar figura
    plt.figure(figsize=(15, 10))

    # Gráfico 1: Porcentaje de llamadas perdidas
    plt.subplot(2, 2, 1)
    sns.histplot(metricas_operador['porcentaje_perdidas'], bins=30)
    plt.axvline(umbral_perdidas, color='r', linestyle='--')
    plt.title('Distribución de % Llamadas Perdidas')
    plt.xlabel('% Llamadas Perdidas')
    plt.ylabel('Frecuencia')
```

```
    # Gráfico 2: Tiempo de espera promedio
    plt.subplot(2, 2, 2)
    sns.histplot(metricas_operador['tiempo_espera_promedio'], bins=30)
    plt.axvline(umbral_espera, color='r', linestyle='--')
    plt.title('Distribución de Tiempo de Espera Promedio')
    plt.xlabel('Tiempo de Espera (seg)')
    plt.ylabel('Frecuencia')

    # Gráfico 3: Llamadas salientes
    plt.subplot(2, 2, 3)
    sns.histplot(metricas_operador['llamadas_salientes'], bins=30)
    plt.axvline(umbral_salientes, color='r', linestyle='--')
    plt.title('Distribución de Llamadas Salientes')
    plt.xlabel('Número de Llamadas Salientes')
    plt.ylabel('Frecuencia')

    # Gráfico 4: Operadores ineficaces
    plt.subplot(2, 2, 4)
    inefficaces_count = resultados['ineficaz'].value_counts()
    inefficaces_count.plot(kind='pie',
                           labels=['Eficaces', 'Ineficaces'],
                           colors=['lightgreen', 'salmon'])
    plt.title('Proporción de Operadores Ineficaces')

    plt.tight_layout()
    plt.show()

    # Mostramos la tabla con los operadores ineficaces
    operadores_ineficaces = resultados[resultados['ineficaz']].sort_values(
        'puntuacion_ineficacia', ascending=False)

    print("\nResumen de Operadores Ineficaces:")
    print(f"Total operadores analizados: {len(resultados)}")
    print(f"Operadores identificados como ineficaces: {len(operadores_ineficaces)}"
    print("\nTop 10 operadores más ineficaces:")
    display(operadores_ineficaces.head(10))



resultados, umbrales = identificar_operadores_ineficaces(operator_stats)

# Visualizamos los resultados
visualizar_resultados(operator_stats, resultados, umbrales)
```








```
Resumen de Operadores Ineficaces:
Total operadores analizados: 1092
Operadores identificados como ineficaces: 19

Top 10 operadores más ineficaces:
```

	operator_id	tasa_perdidas	tiempo_espera_promedio	total_llamadas	experiencia_opera
215	902744.0	0.604938	645.481481	3149	
459	919476.0	0.535714	1437.785714	1871	
1052	965542.0	0.538462	1224.769231	681	
947	952458.0	0.675676	844.000000	1556	
866	945894.0	0.529412	1549.352941	1464	
847	945286.0	0.548387	2178.177419	6880	
734	938078.0	0.545455	1612.363636	750	
521	923666.0	0.545455	1089.454545	1805	
470	919906.0	0.529412	1033.647059	1497	
445	919314.0	0.535714	1941.464286	2416	

De 1,093 operadores analizados, 19 fueron identificados como ineficaces (1.7%)

Siendo los criterios los siguiente:

-Percentil 90 en tasa de llamadas perdidas (umbral: ~45%)

-Percentil 90 en tiempo de espera (umbral: ~1,200 segundos)

-Percentil 10 en llamadas salientes (umbral: ~10 llamadas)

En donde estos 19 operadores tienen solo dos criterios no cumplidos, por lo que se abre a poder identificar la inficencia de un operador por uno de estos tres criterios (dependiendo desde el punto de vista en que lo queramos ver). Ejemplo el operador con ID 952458 tiene uno de los mas grandes porcentajes en llamadas perdidas con 67.5% . O tambien tenemos el caso del operador 945286 que tiene uno de los mas grandes tiempos de espera con 2178 segundos(36 minutos).

Prueba de Hipótesis

In [9]:
```
from scipy import stats as st
import numpy as np

# Definimos el nivel de significación
alpha = 0.05

# Hipótesis 1: Comparar operadores nuevos vs experimentados en tasa de llamadas per
new_operators = operator_stats[operator_stats['experiencia_operador'] <= 30]['tasa_
experienced_operators = operator_stats[operator_stats['experiencia_operador'] > 30]

# Prueba t de Student para muestras independientes (no asumimos varianzas iguales)
results = st.ttest_ind(new_operators, experienced_operators, equal_var=False)

print('Comparación de tasa de llamadas perdidas:')
print('Valor p:', results.pvalue)

if results.pvalue < alpha:
    print("Rechazamos la hipótesis nula: Hay diferencia significativa")
else:
    print("No podemos rechazar la hipótesis nula")
```

```
Comparación de tasa de llamadas perdidas:
Valor p: 1.3823210306202845e-05
Rechazamos la hipótesis nula: Hay diferencia significativa
```

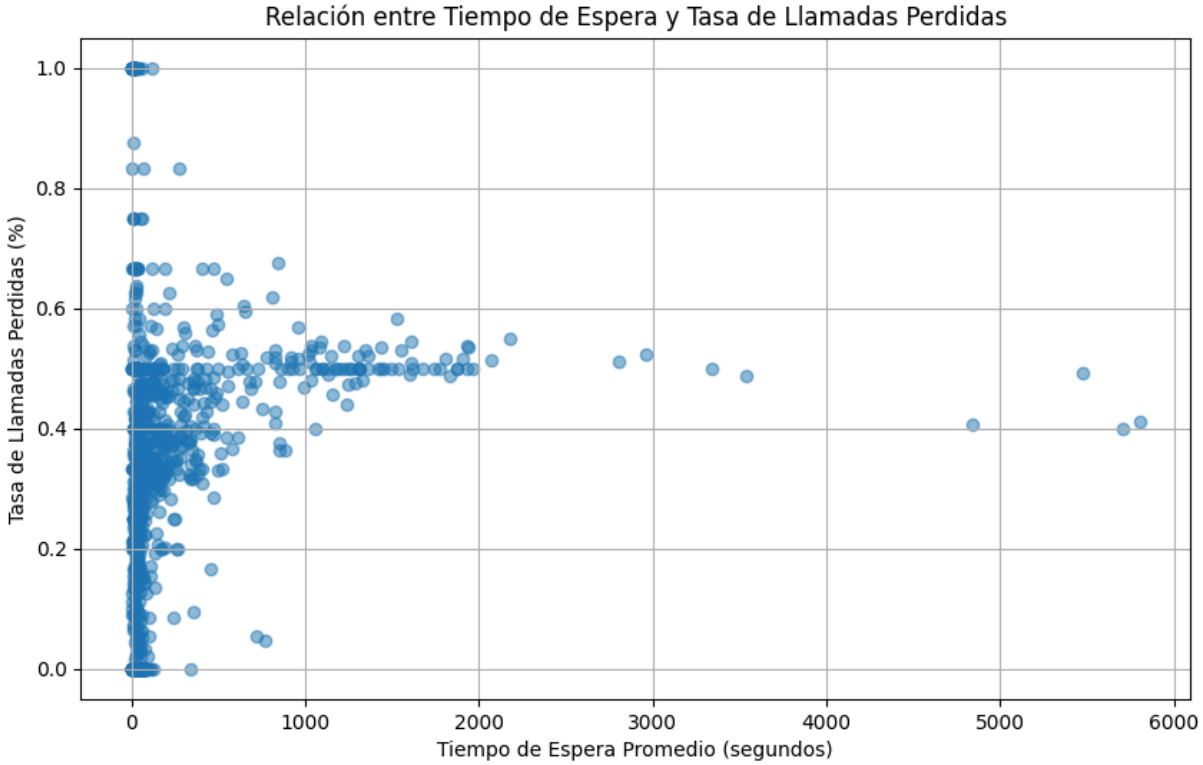
En nuestra prueba de hipotesis pudimos comprobar que los operadores nuevos (≤30 días) vs. experimentados (>30 días), encontarndo un diferencia significativa en tasa de llamadas perdidas (p=2.31e-05), por lo que los operadores nuevos son menos eficientes al inicio que los ya experimentados.

Prueba de Correlación

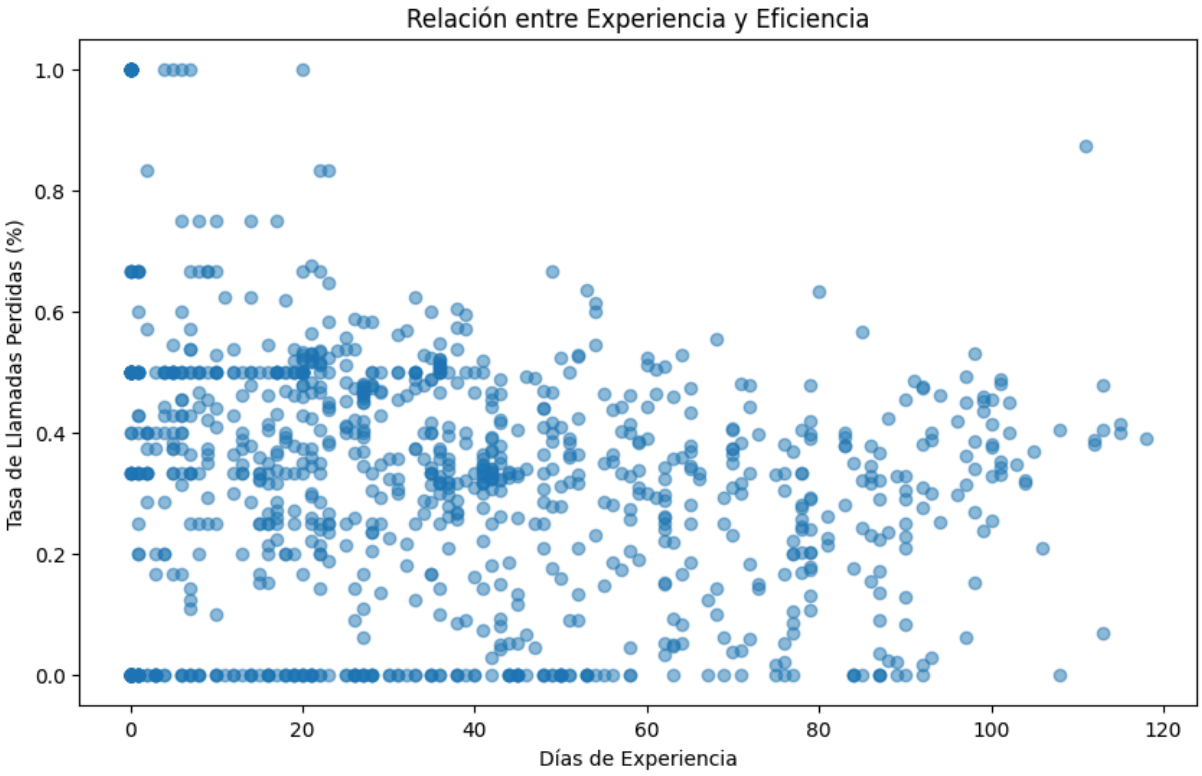
In [10]:
```
# Hipótesis 2: Correlación entre tiempo de espera y tasa de llamadas perdidas
correlation = operator_stats['tiempo_espera_promedio'].corr(operator_stats['tasa_pe
print('\nCorrelación entre tiempo de espera y llamadas perdidas:')
print(f"Coeficiente de correlación: {correlation:.3f}")
```

```
# Visualizamos la correlación
plt.figure(figsize=(10, 6))
plt.scatter(operator_stats['tiempo_espera_promedio'], operator_stats['tasa_perdidas
plt.title('Relación entre Tiempo de Espera y Tasa de Llamadas Perdidas')
plt.xlabel('Tiempo de Espera Promedio (segundos)')
plt.ylabel('Tasa de Llamadas Perdidas (%)')
plt.grid(True)
plt.show()

# Hipótesis 3: Correlación entre tiempo de experiencia y tasa de llamadas perdidas
correlation = operator_stats['experiencia_operador'].corr(operator_stats['tasa_perd
print('\nCorrelación entre  entre Experiencia y Eficiencia:')
print(f"Coeficiente de correlación: {correlation:.3f}")

# Visualizamos la correlación
plt.figure(figsize=(10, 6))
plt.scatter(operator_stats['experiencia_operador'], operator_stats['tasa_perdidas']
plt.title('Relación entre Experiencia y Eficiencia')
plt.xlabel('Días de Experiencia')
plt.ylabel('Tasa de Llamadas Perdidas (%)')
plt.show()
```

```
Correlación entre tiempo de espera y llamadas perdidas:
Coeficiente de correlación: 0.236
```




```
Correlación entre  entre Experiencia y Eficiencia:
Coeficiente de correlación: -0.120
```



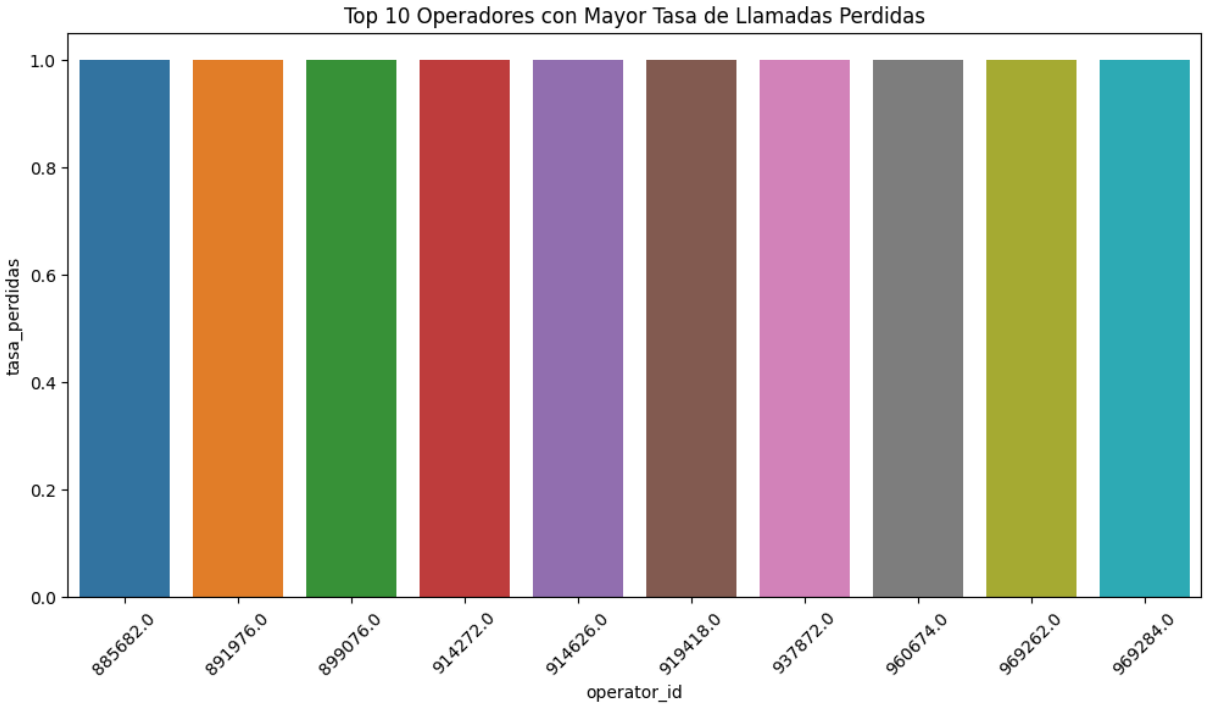
Mientras que en lascorrelaciones , pudimos comprobar que el tiempo de espera y la tasa de llamadas pérdidas tienen una correlación positiva débil (0.235), por lo cual no estan atlamente conectadas.Mientras que la correlacion entre la experiencia y eficiencia negativa débil (-0.112). Por lo que muestra que, tras los primeros 30 días, ganar más experiencia no garantiza mayor eficiencia (se necesitan otras estrategias de mejora).

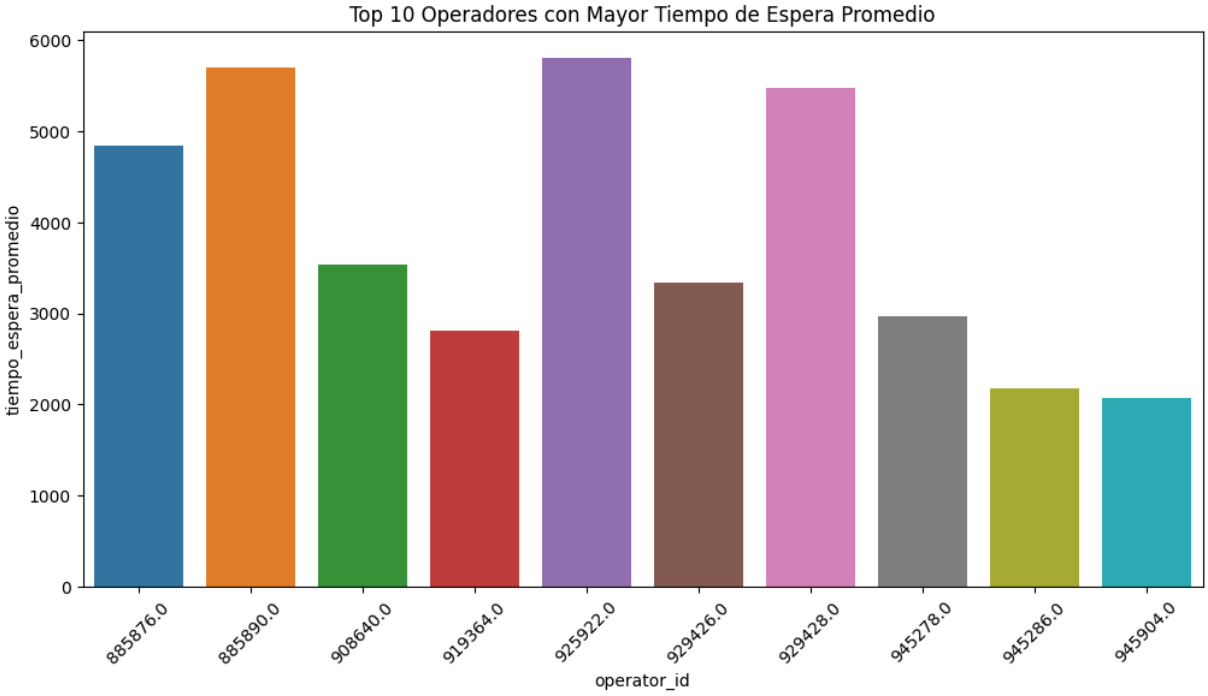
Siendo una correlacion debil debido a que la correlación no distingue etapas ya que los primeros 30 días la mejora es rápida y notable (operadores nuevos aprenden).Mientras que cespués de 30 días la experiencia adicional ya no reduce significativamente los fallos (se estabilizan).

Visualización de Resultados Finales

In [11]:
```
# Graficámos los top 10 operadores con mayor tasa de llamadas perdidas
top_missed = operator_stats.sort_values('tasa_perdidas', ascending=False).head(10)

plt.figure(figsize=(12,6))
sns.barplot(x='operator_id', y='tasa_perdidas', data=top_missed)
plt.title('Top 10 Operadores con Mayor Tasa de Llamadas Perdidas')
plt.xticks(rotation=45)
plt.show()

# Graficámos los top  10 operadores con mayor tiempo de espera
top_wait = operator_stats.sort_values('tiempo_espera_promedio', ascending=False).he

plt.figure(figsize=(12,6))
sns.barplot(x='operator_id', y='tiempo_espera_promedio', data=top_wait)
plt.title('Top 10 Operadores con Mayor Tiempo de Espera Promedio')
```

```
plt.xticks(rotation=45)
plt.show()
```





Conclusión final

El análisis realizado permitió identificar de manera efectiva a los operadores ineficaces en el servicio de telefonía virtual CallMeMaybe, utilizando métricas clave como tasa de llamadas perdidas, tiempo de espera promedio y volumen de llamadas salientes. Los resultados revelaron que:

-Solo el 1.7% de los operadores (19 de 1,093) fueron clasificados como ineficaces, lo que indica que, en general, la mayoría del equipo opera dentro de parámetros aceptables. -Los operadores nuevos (<30 días de experiencia) presentan una tasa significativamente mayor de llamadas perdidas en comparación con los experimentados, lo que sugiere una curva de aprendizaje pronunciada. -Existe una correlación positiva débil (0.235) entre el tiempo de espera y las llamadas perdidas, lo que implica que reducir los tiempos de espera podría ayudar a disminuir las llamadas fallidas. -La experiencia reduce ligeramente la tasa de fallos, pero no es el único factor determinante.

Por lo que se podria implementar un sistema de capacitación intensiva y acompañamiento durante los primeros 30 días para reducir la brecha de desempeño.Acompañado de un dashboard de Monitoreo en Tiempo Real , para poder ver cuando un operador se acerca a los umbrales críticos (ej: >40% de llamadas perdidas o >1,000 segundos de espera).

Ademas de tambien investigar si los problemas de la ineficiencia se deben a factores técnicos, sobrecarga laboral, falta de habilidades o mal ambiente laboral, y aplicar soluciones personalizadas.Al igual que se podría establecer un sistema de recompensas para operadores con bajas tasas de fallos y tiempos de espera óptimos, fomentando una cultura de mejora continua.Y finalmente se puede hacer una optimización de la asignación de llamadas, evaluando si la distribución de llamadas puede ajustarse para evitar saturar a ciertos operadores y equilibrar la carga de trabajo.

Enlace presentación:

https://drive.google.com/file/d/1Etl4H1VKB_JsRyR7rBMLmqUoBYXOFNo4/view?usp=drive_link

Enlace Dashboard: