Predicción de Cancelaciones en Reservas de Hotel

En este notebook se desarrolla un proyecto completo de **Data Science** aplicado a un dataset de un cliente de reservas de hotel.

Los Objetivos principales:

1. **Exploración de datos (EDA):** entender el comportamiento de los clientes, estacionalidad, importante: tasas de cancelación y patrones relevantes. 2. **Limpieza y preparación:** transformar los datos en un formato adecuado para el modelado.
2. **Modelado predictivo:** construir modelos de Machine Learning (Random Forest y XGBoost) para predecir si una reserva será cancelada.
3. **Evaluación de modelos:** comparar rendimiento mediante métricas (accuracy, recall, F1-score, ROC-AUC).
4. **Visualización y reporte:** generar gráficos claros (curva ROC, importancia de variables, matrices de confusión) y un **informe PDF** con resultados y conclusiones.

In [4]:
```
import os
import random
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

%matplotlib inline
plt.rcParams['figure.figsize'] = (10,5)
sns.set_theme(style="whitegrid")

RANDOM_STATE = 42
np.random.seed(RANDOM_STATE)
random.seed(RANDOM_STATE)
```

In [6]:
```
path = r"C:\Users\asiag\Desktop\Hotel.csv"

try:
    df = pd.read_csv(path, low_memory=False)
    print("Leído con utf-8 por defecto.")
except UnicodeDecodeError:
    df = pd.read_csv(path, encoding='latin1', low_memory=False)
    print("Leído con encoding latin1.")
except FileNotFoundError:
    raise FileNotFoundError(f"No encuentro el archivo en: {path}. Revisa la ruta
except Exception as e:
```

```
    raise RuntimeError("Error leyendo el CSV: " + str(e))

print("Shape:", df.shape)
display(df.head(10))
```

```
Leído con utf-8 por defecto.
Shape: (36275, 19)
```

	ID	n_adults	n_children	weekend_nights	week_nights	meal_plan	car_parking_
0	INN00001	2	0	1	2	Meal Plan 1	
1	INN00002	2	0	2	3	Not Selected	
2	INN00003	1	0	2	1	Meal Plan 1	
3	INN00004	2	0	0	2	Meal Plan 1	
4	INN00005	2	0	1	1	Not Selected	
5	INN00006	2	0	0	2	Meal Plan 2	
6	INN00007	2	0	1	3	Meal Plan 1	
7	INN00008	2	0	1	3	Meal Plan 1	
8	INN00009	3	0	0	4	Meal Plan 1	
9	INN00010	2	0	0	5	Meal Plan 1	

In [8]:
```
print("Información general:")
display(df.info())

print("\nEstadísticas numéricas (describe):")
display(df.describe(include='number').T)

print("\nEstadísticas categóricas (describe):")
display(df.describe(include=['object', 'category']).T)

nulls = df.isnull().sum().sort_values(ascending=False)
display(nulls[nulls > 0])

uniques = df.nunique().sort_values()
display(uniques.tail(30))
```

```
Información general:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36275 entries, 0 to 36274
Data columns (total 19 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   ID                              36275 non-null  object
 1   n_adults                        36275 non-null  int64
 2   n_children                      36275 non-null  int64
 3   weekend_nights                  36275 non-null  int64
 4   week_nights                     36275 non-null  int64
 5   meal_plan                       36275 non-null  object
 6   car_parking_space               36275 non-null  int64
 7   room_type                       36275 non-null  object
 8   lead_time                       36275 non-null  int64
 9   year                            36275 non-null  int64
 10  month                           36275 non-null  int64
 11  date                            36275 non-null  int64
 12  market_segment                  36275 non-null  object
 13  repeated_guest                  36275 non-null  int64
 14  previous_cancellations          36275 non-null  int64
 15  previous_bookings_not_canceled  36275 non-null  int64
 16  avg_room_price                  36275 non-null  float64
 17  special_requests                36275 non-null  int64
 18  status                          36275 non-null  object
dtypes: float64(1), int64(13), object(5)
memory usage: 5.3+ MB
None
Estadísticas numéricas (describe):
```

	count	mean	std	min	25%	50%
n_adults	36275.0	1.844962	0.518715	0.0	2.0	2.00
n_children	36275.0	0.105279	0.402648	0.0	0.0	0.00
weekend_nights	36275.0	0.810724	0.870644	0.0	0.0	1.00
week_nights	36275.0	2.204300	1.410905	0.0	1.0	2.00
car_parking_space	36275.0	0.030986	0.173281	0.0	0.0	0.00
lead_time	36275.0	85.232557	85.930817	0.0	17.0	57.00
year	36275.0	2017.820427	0.383836	2017.0	2018.0	2018.00
month	36275.0	7.423653	3.069894	1.0	5.0	8.00
date	36275.0	15.596995	8.740447	1.0	8.0	16.00
repeated_guest	36275.0	0.025637	0.158053	0.0	0.0	0.00
previous_cancellations	36275.0	0.023349	0.368331	0.0	0.0	0.00
previous_bookings_not_canceled	36275.0	0.153411	1.754171	0.0	0.0	0.00
avg_room_price	36275.0	103.423539	35.089424	0.0	80.3	99.45
special_requests	36275.0	0.619655	0.786236	0.0	0.0	0.00

```
Estadísticas categóricas (describe):
```

	count	unique	top	freq
ID	36275	36275	INN00001	1
meal_plan	36275	4	Meal Plan 1	27835
room_type	36275	7	Room_Type 1	28130
market_segment	36275	5	Online	23214
status	36275	2	Not_Canceled	24390

```
Series([], dtype: int64)
car_parking_space                     2
year                                  2
repeated_guest                        2
status                                2
meal_plan                             4
n_adults                              5
market_segment                        5
n_children                            6
special_requests                      6
room_type                             7
weekend_nights                        8
previous_cancellations                9
month                                12
week_nights                          18
date                                 31
previous_bookings_not_canceled       59
lead_time                           352
avg_room_price                     3930
ID                                36275
dtype: int64
```

In [10]:
```
possible_date_cols = [c for c in df.columns if any(k in c.lower() for k in ['dat
print("Columnas candidatas a fecha:", possible_date_cols)

for c in possible_date_cols:

    df[c] = pd.to_datetime(df[c], errors='coerce')

for c in possible_date_cols:
    non_na = df[c].notna().sum()
    print(f"{c}: {non_na} valores parseados como fecha")
```

```
Columnas candidatas a fecha: ['date']
date: 36275 valores parseados como fecha
```

In [12]:
```
possible_targets = [c for c in df.columns if c.lower() in ('is_canceled', 'cance
print("Targets detectados:", possible_targets)

if 'is_canceled' in df.columns:

    df['is_canceled'] = df['is_canceled'].astype(int)
    cancel_rate = df['is_canceled'].mean()
    print(f"Tasa global de cancelación: {cancel_rate:.2%} ({df['is_canceled'].su

num_cols = df.select_dtypes(include=['number']).columns.tolist()
cat_cols = df.select_dtypes(include=['object', 'category']).columns.tolist()
print(f"Num cols ({len(num_cols)}): {num_cols[:10]} ...")
print(f"Cat cols ({len(cat_cols)}): {cat_cols[:10]} ...")
```

```
Targets detectados: []
Num cols (13): ['n_adults', 'n_children', 'weekend_nights', 'week_nights', 'car_p
arking_space', 'lead_time', 'year', 'month', 'repeated_guest', 'previous_cancella
tions'] ...
Cat cols (5): ['ID', 'meal_plan', 'room_type', 'market_segment', 'status'] ...
```

In [14]:
```
if set(['stays_in_weekend_nights', 'stays_in_week_nights']).issubset(df.columns)
    df['total_nights'] = df['stays_in_weekend_nights'].fillna(0) + df['stays_in_
    print("Creada columna total_nights.")

if set(['adults', 'children', 'babies']).issubset(df.columns):
    df['total_guests'] = df['adults'].fillna(0) + df['children'].fillna(0) + df[
    print("Creada columna total_guests.")

if 'adr' in df.columns and 'total_nights' in df.columns:
    df['estimated_revenue'] = df['adr'].astype(float).fillna(0) * df['total_nigh
    print("Creada columna estimated_revenue (adr * total_nights).")
```

In [16]:
```
msno.matrix(df.sample(min(1000, len(df))))  # sample si dataset grande
plt.title("Missing values (muestra)")

if 'adr' in df.columns:
    plt.figure(figsize=(10,4))
    sns.histplot(df['adr'].dropna(), bins=50, kde=True)
    plt.title("Distribución de ADR (average daily rate)")
    plt.xlabel("ADR")
    plt.show()

if 'is_canceled' in df.columns:
    if 'arrival_date' in df.columns and df['arrival_date'].notna().sum() > 0:
        by_month = (df.groupby(df['arrival_date'].dt.to_period('M'))['is_cancele
        plt.figure(figsize=(12,4))
        by_month.plot(marker='o')
        plt.ylabel("Tasa de cancelación")
        plt.title("Tasa de cancelación por mes")
        plt.show()
    else:
        print("No hay columna arrival_date parseada; omito gráfico mensual.")
```

In [18]:
```
id_cols = [c for c in df.columns if 'id' in c.lower()]
print("Posibles columnas ID:", id_cols)
for c in id_cols:
    print(c, "duplicados?", df[c].duplicated().sum())

if 'country' in df.columns:
    display(df['country'].value_counts().head(20))

if 'adr' in df.columns:
    print("ADR - percentiles:")
    print(df['adr'].describe(percentiles=[.01,.05,.25,.5,.75,.95,.99]))
```

```
Posibles columnas ID: ['ID']
ID duplicados? 0
```

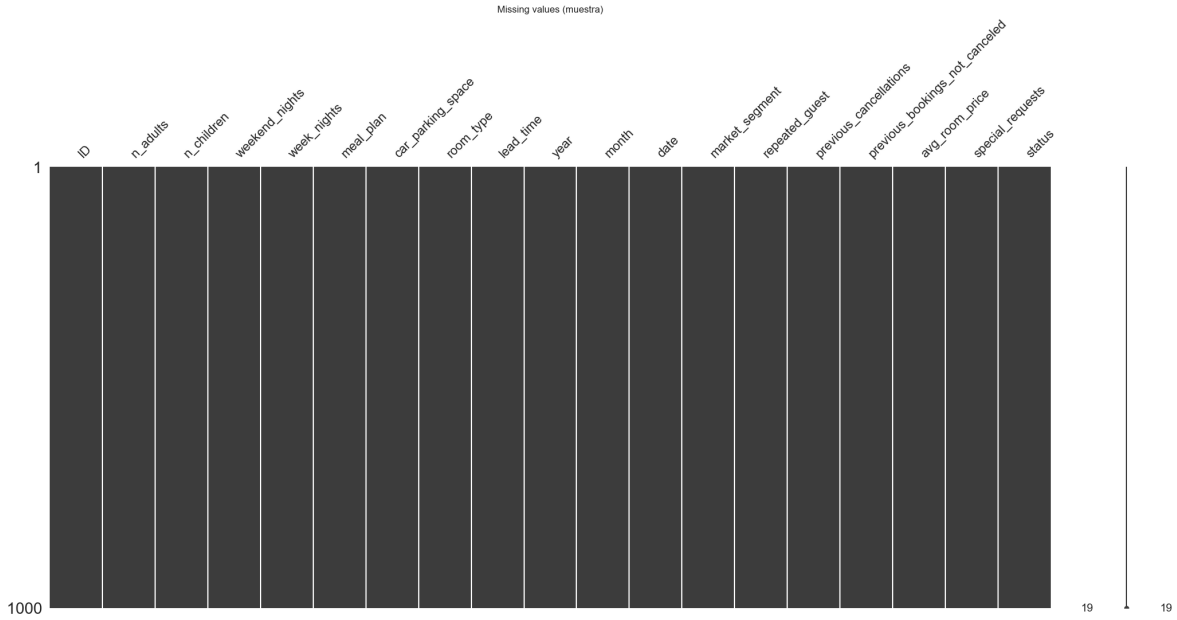
In [21]:
```
df_clean = df.copy()

df_clean['is_canceled'] = df_clean['status'].apply(lambda x: 1 if x == 'Canceled

df_clean['total_nights'] = df_clean['weekend_nights'] + df_clean['week_nights']
df_clean['total_guests'] = df_clean['n_adults'] + df_clean['n_children']
```

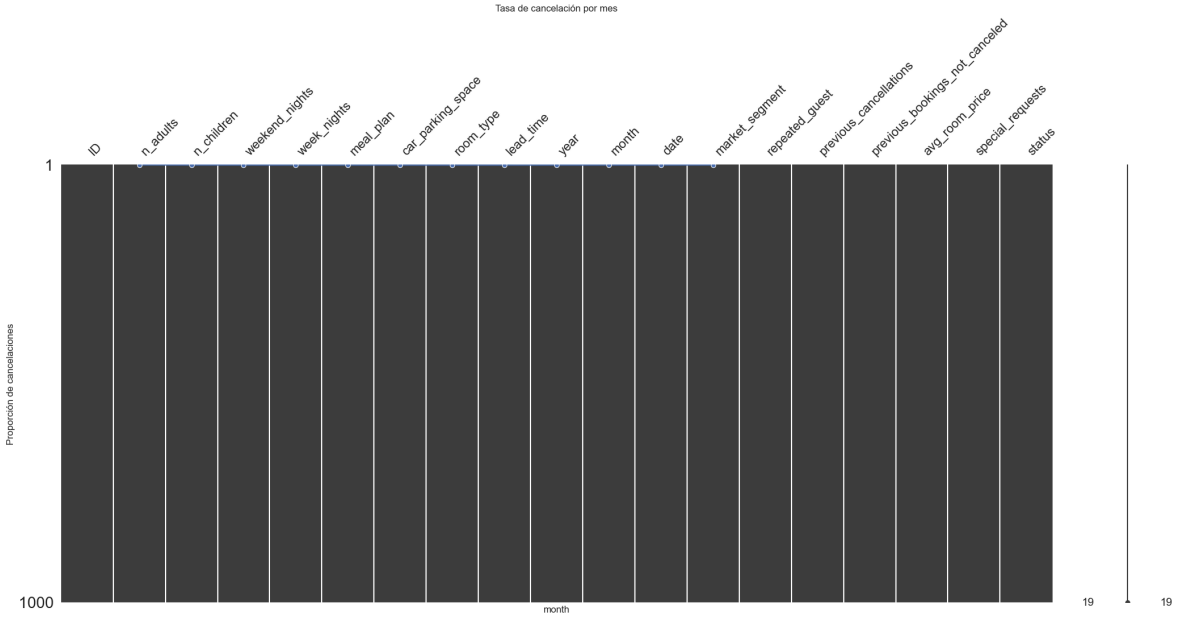
```
df_clean['check_in_date'] = pd.to_datetime(
    df_clean[['year', 'month', 'date']].rename(columns={'year':'year','month':'m
    errors='coerce'
)

q99 = df_clean['avg_room_price'].quantile(0.99)
df_clean = df_clean[df_clean['avg_room_price'] < q99]  # recorte al percentil 99
print(f"Filtrado precios > P99 ({q99:.2f}). Nuevo shape: {df_clean.shape}")

print(df_clean[['is_canceled','total_nights','total_guests','avg_room_price']].d
```

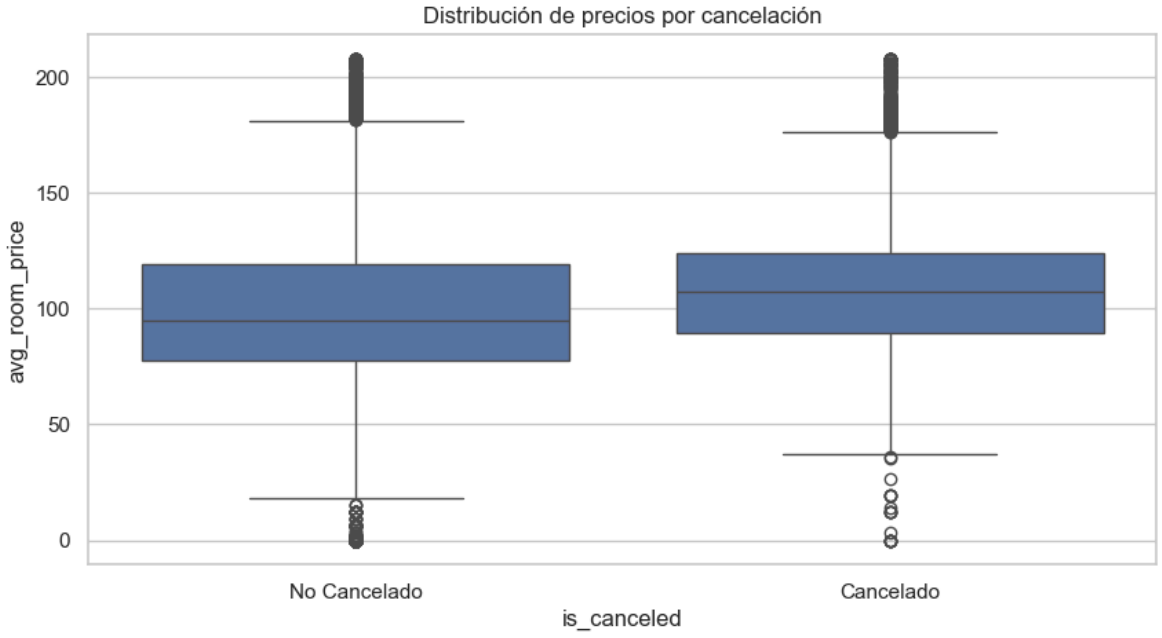
```
Filtrado precios > P99 (208.00). Nuevo shape: (35909, 23)
        is_canceled  total_nights  total_guests  avg_room_price
count  35909.000000  35909.000000  35909.000000    35909.000000
mean       0.325629      3.015400      1.935726      102.095498
std        0.468616      1.785355      0.630658       32.554089
min        0.000000      0.000000      1.000000        0.000000
25%        0.000000      2.000000      2.000000       80.100000
50%        0.000000      3.000000      2.000000       99.200000
75%        1.000000      4.000000      2.000000      120.000000
max        1.000000     24.000000     12.000000      207.900000
```

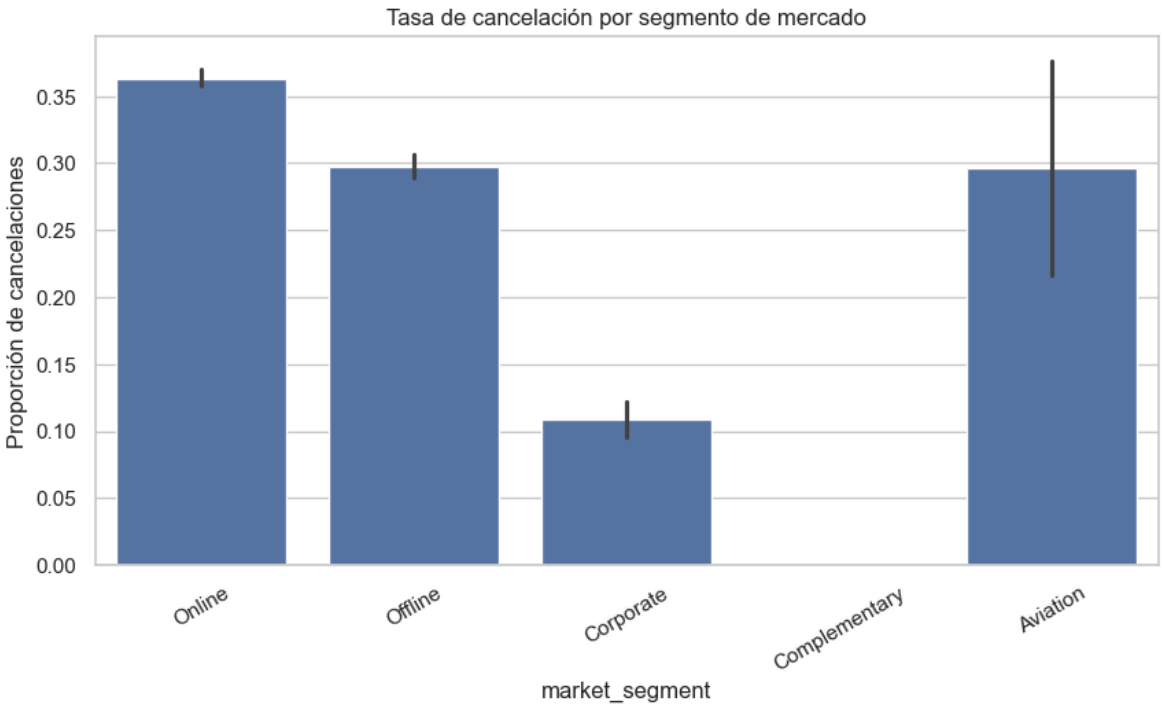
In [22]:
```
cancel_rate = df_clean['is_canceled'].mean()
print(f"Tasa global de cancelación: {cancel_rate:.1%}")

cancel_by_month = df_clean.groupby('month')['is_canceled'].mean()
sns.lineplot(x=cancel_by_month.index, y=cancel_by_month.values, marker='o')
plt.title("Tasa de cancelación por mes")
plt.ylabel("Proporción de cancelaciones")
plt.show()

sns.boxplot(data=df_clean, x='is_canceled', y='avg_room_price')
plt.title("Distribución de precios por cancelación")
plt.xticks([0,1], ["No Cancelado","Cancelado"])
plt.show()

sns.barplot(
    data=df_clean,
    x='market_segment',
    y='is_canceled',
    estimator=np.mean,
    order=df_clean['market_segment'].value_counts().index
)
plt.title("Tasa de cancelación por segmento de mercado")
plt.ylabel("Proporción de cancelaciones")
plt.xticks(rotation=30)
plt.show()

sns.histplot(df_clean['total_nights'], bins=30, kde=False)
plt.title("Distribución de número de noches")
plt.show()
```

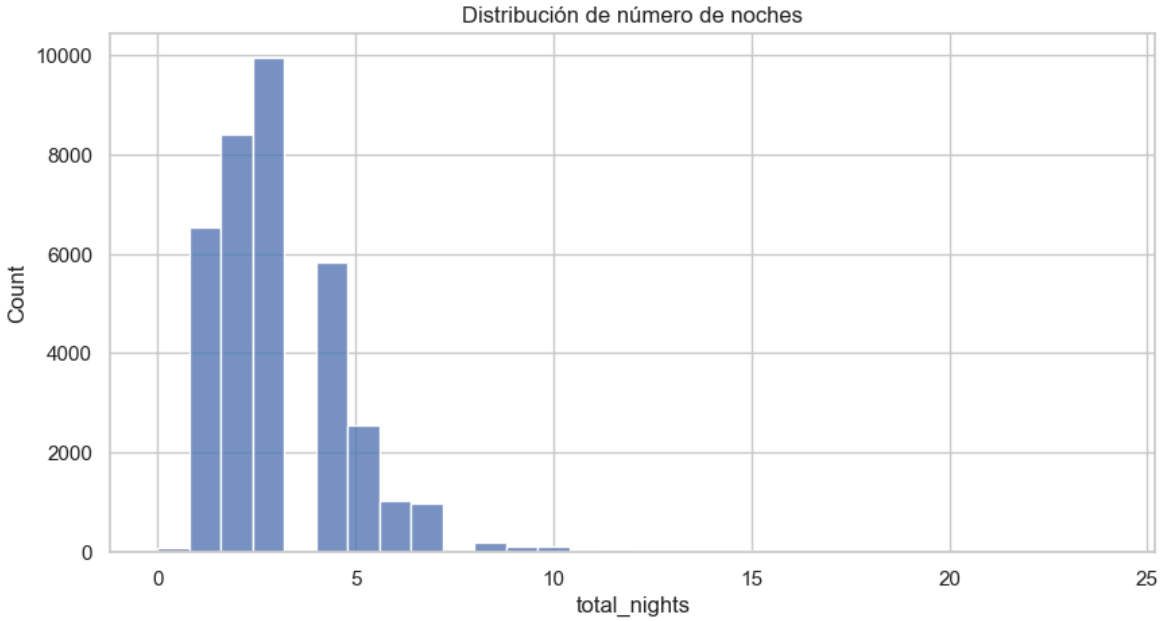
```
Tasa global de cancelación: 32.6%
```


Missing values (muestra)
ID
n_adults
n_children
weekend_nights
week_nights
meal_plan
car_parking_space
room_type
lead_time
year
month
date
market_segment
repeated_guest
previous_cancellations
previous_bookings_not_canceled
avg_room_price
special_requests
status
1
1000
19
19


Tasa de cancelación por mes
ID
n_adults
n_children
weekend_nights
week_nights
meal_plan
car_parking_space
room_type
lead_time
year
month
date
market_segment
repeated_guest
previous_cancellations
previous_bookings_not_canceled
avg_room_price
special_requests
status
Proporción de cancelaciones
1
1000
month
19
19


Distribución de precios por cancelación
avg_room_price
200
150
100
50
0
No Cancelado
Cancelado
is_canceled

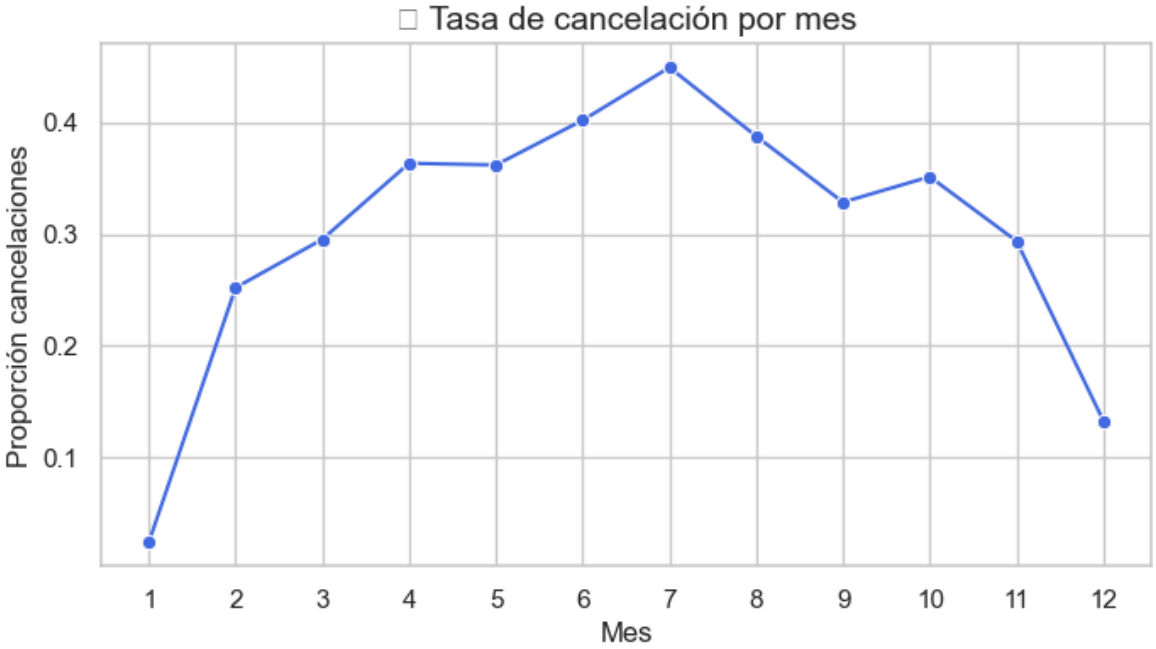




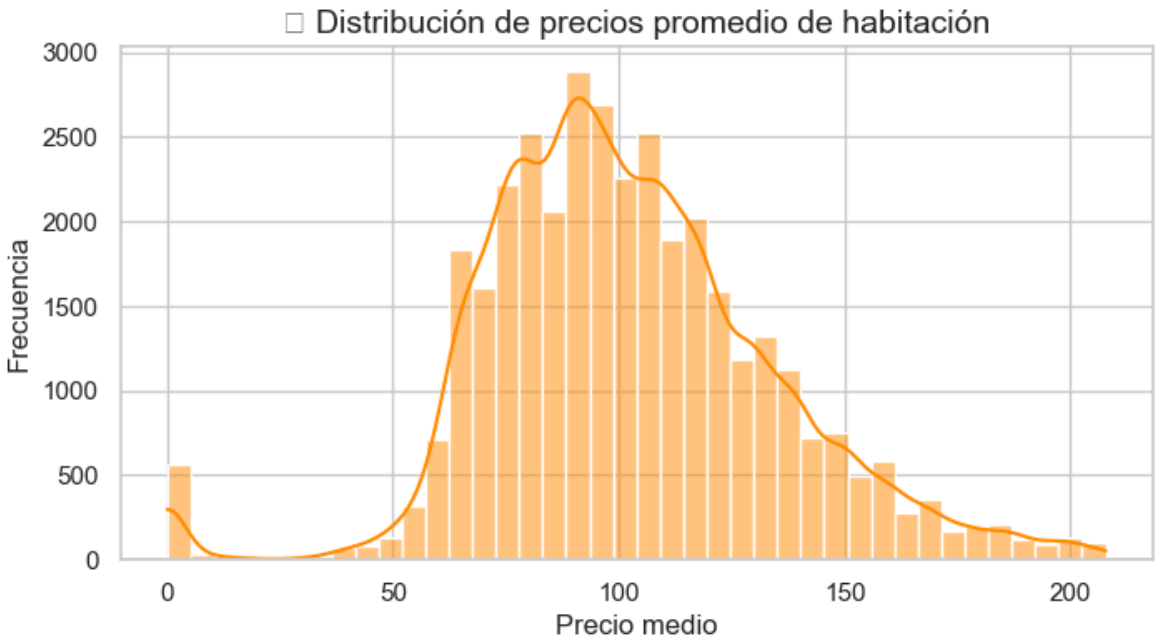
In [24]:
```
cancel_by_month = df_clean.groupby('month')['is_canceled'].mean().reset_index()

plt.figure(figsize=(8,4))
sns.lineplot(
    data=cancel_by_month,
    x='month',
    y='is_canceled',
    marker='o',
    color='royalblue'
)
plt.xticks(range(1,13))
plt.title("📅 Tasa de cancelación por mes", fontsize=14)
plt.ylabel("Proporción cancelaciones")
plt.xlabel("Mes")
plt.show()


plt.figure(figsize=(8,4))
sns.histplot(df_clean['avg_room_price'], bins=40, kde=True, color="darkorange")
```

```
plt.title("💰 Distribución de precios promedio de habitación", fontsize=14)
plt.xlabel("Precio medio")
plt.ylabel("Frecuencia")
plt.show()
```

```
C:\Users\asiag\anaconda3\envs\NEXUM\Lib\site-packages\IPython\core\pylabtools.py:
170: UserWarning: Glyph 128197 (\N{CALENDAR}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```




```
C:\Users\asiag\anaconda3\envs\NEXUM\Lib\site-packages\IPython\core\pylabtools.py:
170: UserWarning: Glyph 128176 (\N{MONEY BAG}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [36]:
```
drop_cols = ['ID','status','is_canceled','check_in_date']
X = df_clean.drop(columns=drop_cols, errors='ignore')
y = df_clean['is_canceled']

X = pd.get_dummies(X, drop_first=True)

X = X.select_dtypes(include=['int64','float64','uint8','bool'])
```

```
print("Tipos de X tras limpieza:")
print(X.dtypes.value_counts())
```

```
Tipos de X tras limpieza:
int64      14
bool       13
float64     1
Name: count, dtype: int64
```

In [ ]:

In [28]:
```
print("Columnas no numéricas en X:", X.select_dtypes(exclude=['int64','float64',
```

```
Columnas no numéricas en X: []
```

In [38]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, roc_auc_score, roc_curve, con
import matplotlib.pyplot as plt
import seaborn as sns

df_clean['is_canceled'] = df_clean['status'].apply(lambda x: 1 if x == 'Canceled

y = df_clean['is_canceled']
X = df_clean.drop(columns=['ID','status','is_canceled','check_in_date'], errors=

num_cols = X.select_dtypes(include=['int64','float64']).columns.tolist()
cat_cols = X.select_dtypes(include=['object','category']).columns.tolist()

print("Numéricas:", num_cols)
print("Categóricas:", cat_cols)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=RANDOM_STATE
)
```

```
Numéricas: ['n_adults', 'n_children', 'weekend_nights', 'week_nights', 'car_parki
ng_space', 'lead_time', 'year', 'month', 'repeated_guest', 'previous_cancellation
s', 'previous_bookings_not_canceled', 'avg_room_price', 'special_requests', 'tota
l_nights', 'total_guests']
Categóricas: ['meal_plan', 'room_type', 'market_segment']
```

In [39]:
```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), num_cols),
        ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), cat_cols)
    ]
)
```

In [40]:
```
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

rf_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', RandomForestClassifier(
        n_estimators=300,
        max_depth=12,
        random_state=RANDOM_STATE,
        n_jobs=-1
```

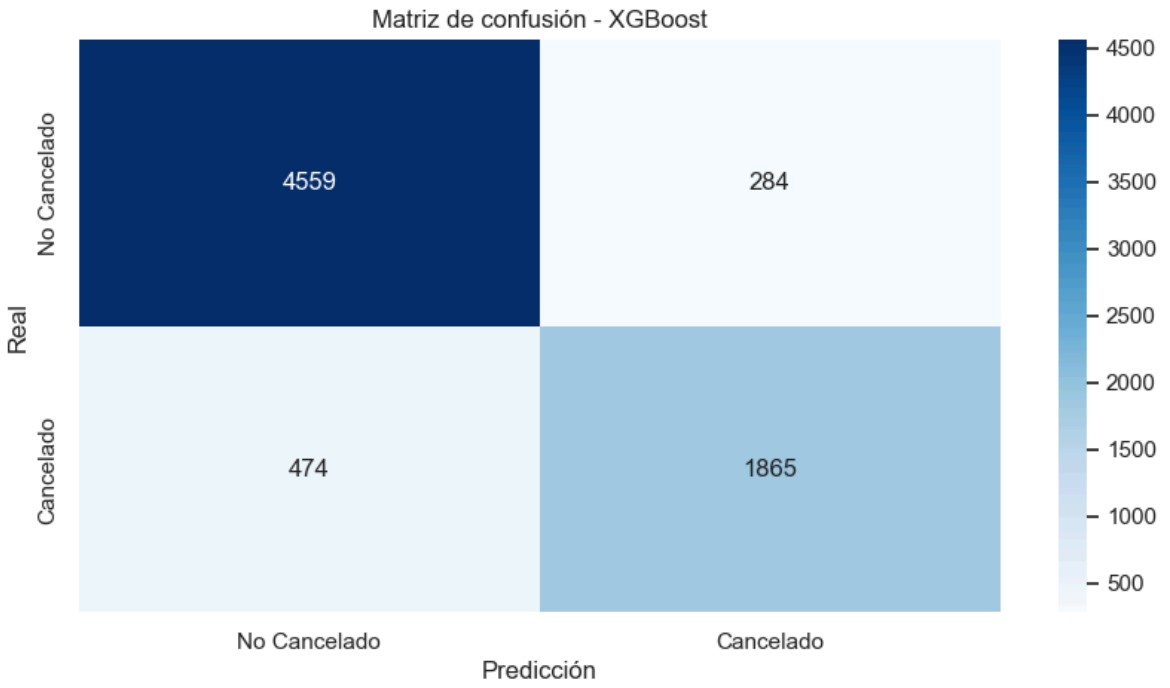
```
    ))
])

rf_pipeline.fit(X_train, y_train)
y_pred_rf = rf_pipeline.predict(X_test)
y_proba_rf = rf_pipeline.predict_proba(X_test)[:,1]

print("=== Random Forest ===")
print(classification_report(y_test, y_pred_rf))
print("ROC-AUC:", roc_auc_score(y_test, y_proba_rf))

xgb_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', XGBClassifier(
        n_estimators=400,
        max_depth=8,
        learning_rate=0.1,
        subsample=0.8,
        colsample_bytree=0.8,
        random_state=RANDOM_STATE,
        use_label_encoder=False,
        eval_metric='logloss'
    ))
])

xgb_pipeline.fit(X_train, y_train)
y_pred_xgb = xgb_pipeline.predict(X_test)
y_proba_xgb = xgb_pipeline.predict_proba(X_test)[:,1]

print("=== XGBoost ===")
print(classification_report(y_test, y_pred_xgb))
print("ROC-AUC:", roc_auc_score(y_test, y_proba_xgb))
```

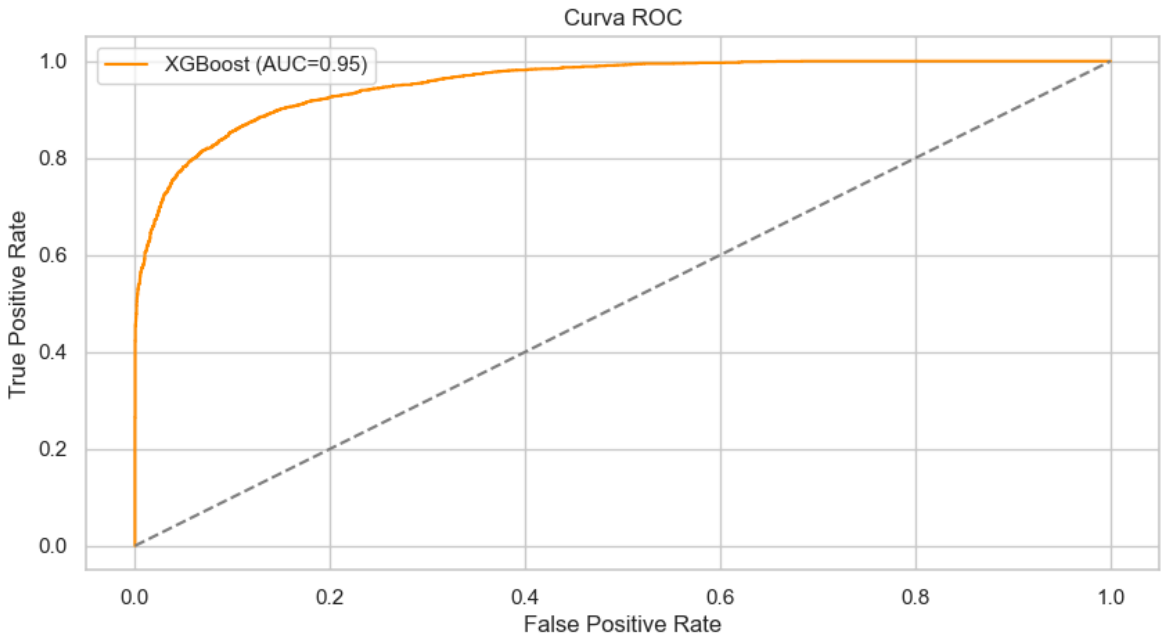
```
=== Random Forest ===
              precision    recall  f1-score   support

           0       0.89      0.94      0.91      4843
           1       0.86      0.75      0.80      2339

    accuracy                           0.88      7182
   macro avg       0.87      0.85      0.86      7182
weighted avg       0.88      0.88      0.88      7182

ROC-AUC: 0.9374147284149397
```

```
C:\Users\asiag\anaconda3\envs\NEXUM\Lib\site-packages\xgboost\training.py:183: Us
erWarning: [12:29:57] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learne
r.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

```
=== XGBoost ===
              precision    recall  f1-score   support

           0       0.91      0.94      0.92      4843
           1       0.87      0.80      0.83      2339

    accuracy                           0.89      7182
   macro avg       0.89      0.87      0.88      7182
weighted avg       0.89      0.89      0.89      7182

ROC-AUC: 0.9539741116019498
```

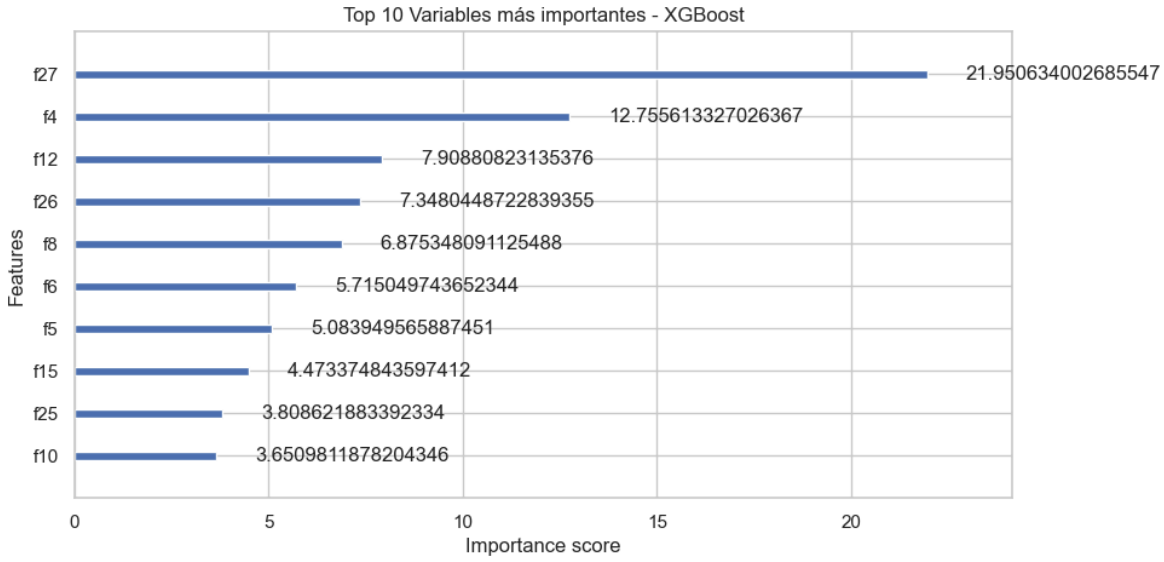
In [41]:
```
cm = confusion_matrix(y_test, y_pred_xgb)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["No Cancelado","
plt.title("Matriz de confusión - XGBoost")
plt.ylabel("Real")
plt.xlabel("Predicción")
plt.show()

fpr, tpr, _ = roc_curve(y_test, y_proba_xgb)
plt.plot(fpr, tpr, label=f"XGBoost (AUC={roc_auc_score(y_test, y_proba_xgb):.2f}
plt.plot([0,1],[0,1], linestyle="--", color="grey")
plt.title("Curva ROC")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```





In [42]:
```
from xgboost import plot_importance

plt.figure(figsize=(10,6))
plot_importance(xgb_pipeline.named_steps['model'], max_num_features=10, importan
plt.title("Top 10 Variables más importantes - XGBoost")
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



📌 Conclusiones y Recomendaciones

- Se desarrolló un **modelo predictivo de cancelaciones** usando Random Forest y XGBoost, con **XGBoost obteniendo el mejor desempeño** (Accuracy 0.89, ROC-AUC 0.95).
- Variables más influyentes: `lead_time`, `market_segment`, `previous_cancellations`, `avg_room_price` y tipo de habitación.
- El modelo permite al hotel **anticipar reservas con alta probabilidad de cancelación**, optimizando la asignación de habitaciones y reduciendo pérdidas por cancelaciones de último minuto.

- El pipeline incluye **EDA, limpieza, modelado, visualización y generación de un informe PDF**, mostrando un flujo completo de Data Science profesional.

Recomendaciones para el hotel:

1. Implementar políticas de prepago o confirmación para reservas con alto riesgo de cancelación.
2. Optimizar la asignación de habitaciones según la probabilidad de cancelación.
3. Continuar monitoreando y actualizando el modelo con nuevos datos para mantener su precisión.