# ¿Qué vende un coche?

Eres un analista en Crankshaft List. Cientos de anuncios gratuitos de vehículos se publican en tu sitio web cada día. Necesitas estudiar los datos recopilados durante los últimos años y determinar qué factores influyen en el precio de un vehículo.

**Respuesta del estudiante** El proyecto sera orientado en descubrir que factores influyen en el precio de los vehiculos publicados en el sitio web, en el proyecto dependiendo de lo que podamos encontrar en la data, tendremos que evaluar los datos que nos han entregado, corregir las variables, valores nulos, datos atipicos, estudiar la correlacion que hay entre las columnas que existen en el dataframe, y verificar los indicadores que nos permiten evaluar correctamente la premisa principal que es saber que factores influyen en la variabilidad de los precios de los vehiculos publicados, luego de las correcciones de lugar estaremos explicando las hipotesis que nos llevaran a la conclusion de que cosas influyen en estos cambios de precio, para poder tener una idea mas clara de como ofrecer un producto interesante a los clientes que tenemos en la pagina web, como ejemplo podriamos ofrecer un premium fee para los clientes de vehiculos mas nuevos que quieran publicarlos de manera mas visible en la pagina.

## Inicialización

In [1]:

```
# Cargar todas las librerías

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

### Cargar datos

In [2]:

```
# Carga el archivo de datos en un DataFrame

try:
    df = pd.read_csv('vehicles_us.csv')
except:
    df = pd.read_csv('/datasets/vehicles_us.csv')
```

### Explorar datos iniciales

El dataset contiene los siguientes campos:

- `price`
- `model_year`
- `model`
- `condition`
- `cylinders`

- `fuel` — gasolina, diesel, etc.
- `odometer` — el millaje del vehículo cuando el anuncio fue publicado
- `transmission`
- `paint_color`
- `is_4wd` — si el vehículo tiene tracción a las 4 ruedas (tipo Booleano)
- `date_posted` — la fecha en la que el anuncio fue publicado
- `days_listed` — desde la publicación hasta que se elimina

In [3]:
```
# imprime la información general/resumida sobre el DataFrame

df.shape
```

Out[3]: (51525, 13)

In [4]:
```
# imprimir una muestra de datos

df.head(10)
```

Out[4]:

| | price | model_year | model | condition | cylinders | fuel | odometer | transmission | type | paint_color |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 9400 | 2011.0 | bmw x5 | good | 6.0 | gas | 145000.0 | automatic | SUV | NaN |
| **1** | 25500 | NaN | ford f-150 | good | 6.0 | gas | 88705.0 | automatic | pickup | white |
| **2** | 5500 | 2013.0 | hyundai sonata | like new | 4.0 | gas | 110000.0 | automatic | sedan | red |
| **3** | 1500 | 2003.0 | ford f-150 | fair | 8.0 | gas | NaN | automatic | pickup | NaN |
| **4** | 14900 | 2017.0 | chrysler 200 | excellent | 4.0 | gas | 80903.0 | automatic | sedan | black |
| **5** | 14990 | 2014.0 | chrysler 300 | excellent | 6.0 | gas | 57954.0 | automatic | sedan | black |
| **6** | 12990 | 2015.0 | toyota camry | excellent | 4.0 | gas | 79212.0 | automatic | sedan | white |
| **7** | 15990 | 2013.0 | honda pilot | excellent | 6.0 | gas | 109473.0 | automatic | SUV | black |
| **8** | 11500 | 2012.0 | kia sorento | excellent | 4.0 | gas | 104174.0 | automatic | SUV | NaN |
| **9** | 9200 | 2008.0 | honda pilot | excellent | NaN | gas | 147191.0 | automatic | SUV | blue |

**Respuesta del estudiante** Por aca podemos ver un DataFrame de 13 columnas el cual contiene la informacion necesaria para la investigacion, segun puedo observar hay varios detalles, valores ausentes $NaN$, la columna $\mod el_year$ usa valores con decimales, en vez de enteros, la columna $is_4wd$, deberia usar datos de tipo booleano.

**Respuesta del estudiante** Hay columnas con datos inapropiados en el DataFrame, las columnas con datos inapropiados puedo decir que estan en $\mod el_y ear$ y sus valores decimales, $is_4 wd$ con valores decimales en vez de booleanos.

**Respuesta del estudiante** Hay valores ausentes en algunas columnas, las columnas con mayor incidencia son $\mod el_y ear, cyl \in ders, odometer, pa\int$ y $is_4 wd$. A primera vista veo unos datos que no estan tan mal, donde Sí, debemos corregir unas cuantas cosas como los valores ausentes y los tipos de variables erroneas, tambien demostrar que los datos al ser tratados mantienen la coherencia y son viables para la investigacion.

## Conclusiones y siguientes pasos

**Respuesta del estudiante** Entiendo que hay varias columnas que parecen normales, y a primera vista podrian usarse asi mismo, como son: $price, \mod el, condition, fuel, transmission, type, date_p osted$ y $days_l isted$, por otro lado tenemos las columnas con anomalias como: $\mod el_y ear, cyl \in ders, odometer, pa\int, is_4 wd$, que deben ser evaluadas y corregidas para poder utilizar su data en la investigacion.

# Tratar los valores ausentes (si hay)

**Respuesta del estudiante** Empezaremos por tratar los valores ausentes columna por columna, ya que cada una necesita un tratamiento ligeramente diferente, empecemos con $\mod el_y ear$, $cyl \in ders$ y $odometer$.

In [5]:

```
# verificar la media y la mediana de 'model_year'

mean_model_year = df['model_year'].mean()
median_model_year = df['model_year'].median()

nan_count_model_year = df['model_year'].isna().sum()

print("Cantidad de valores NaN en model_year:", nan_count_model_year)

print("Media de model_year:", mean_model_year)
print("Mediana de model_year:", median_model_year)
```

```
Cantidad de valores NaN en model_year: 3619
Media de model_year: 2009.75046966977
Mediana de model_year: 2011.0
```

In [6]:

```
# eliminar los valores ausentes de 'model_year'

df['model_year'] = df['model_year'].fillna(median_model_year)
```

**Respuesta del estudiante** Luego de observar que la cantidad de valores Nulos de la columna $\mathrm{mod}\ el_y ear$ es solo un $7.02\ \%$ decidi que dado que la media para completar esos valores no nos conviene porque da un valor con decimales, la mediana si nos convendria, ya que su output es un numero entero y variaria poco los resultados finales a pesar de todo, solo habria que abordar el tipo de variable pasandola a $\int 64$, lo cual haremos un poco mas adelante.

In [7]:

```
# verificar la media y la mediana de 'cylinders'

mean_cylinders = df['cylinders'].mean()
median_cylinders = df['cylinders'].median()

nan_count_cylinders = df['cylinders'].isna().sum()

print("Cantidad de valores NaN en cylinders:", nan_count_cylinders)

print("Media de cylinders:", mean_cylinders)
print("Mediana de cylinders:", median_cylinders)
```

```
Cantidad de valores NaN en cylinders: 5260
Media de cylinders: 6.125235058899817
Mediana de cylinders: 6.0
```

In [8]:

```
# eliminar los valores ausentes de 'cylinders'

df['cylinders'] = df['cylinders'].fillna(median_cylinders)
```

**Respuesta del estudiante** Dado que las observaciones en la columna $cyl \in ders$ van en la misma linea que la anterior, y la cantidad de valores Nulos de la columna es de un $10.20\ \%$ , y viendo que dado que la media para completar esos valores nos da un valor con decimales, la mediana seria la mejor opcion.

In [9]:

```
# verificar la media y la mediana de 'odometer'

mean_odometer = df['odometer'].mean()
median_odometer = df['odometer'].median()

nan_count_odometer = df['odometer'].isna().sum()

print("Cantidad de valores NaN en odometer:", nan_count_odometer)

print("Media de odometer:", mean_odometer)
print("Mediana de odometer:", median_odometer)
```

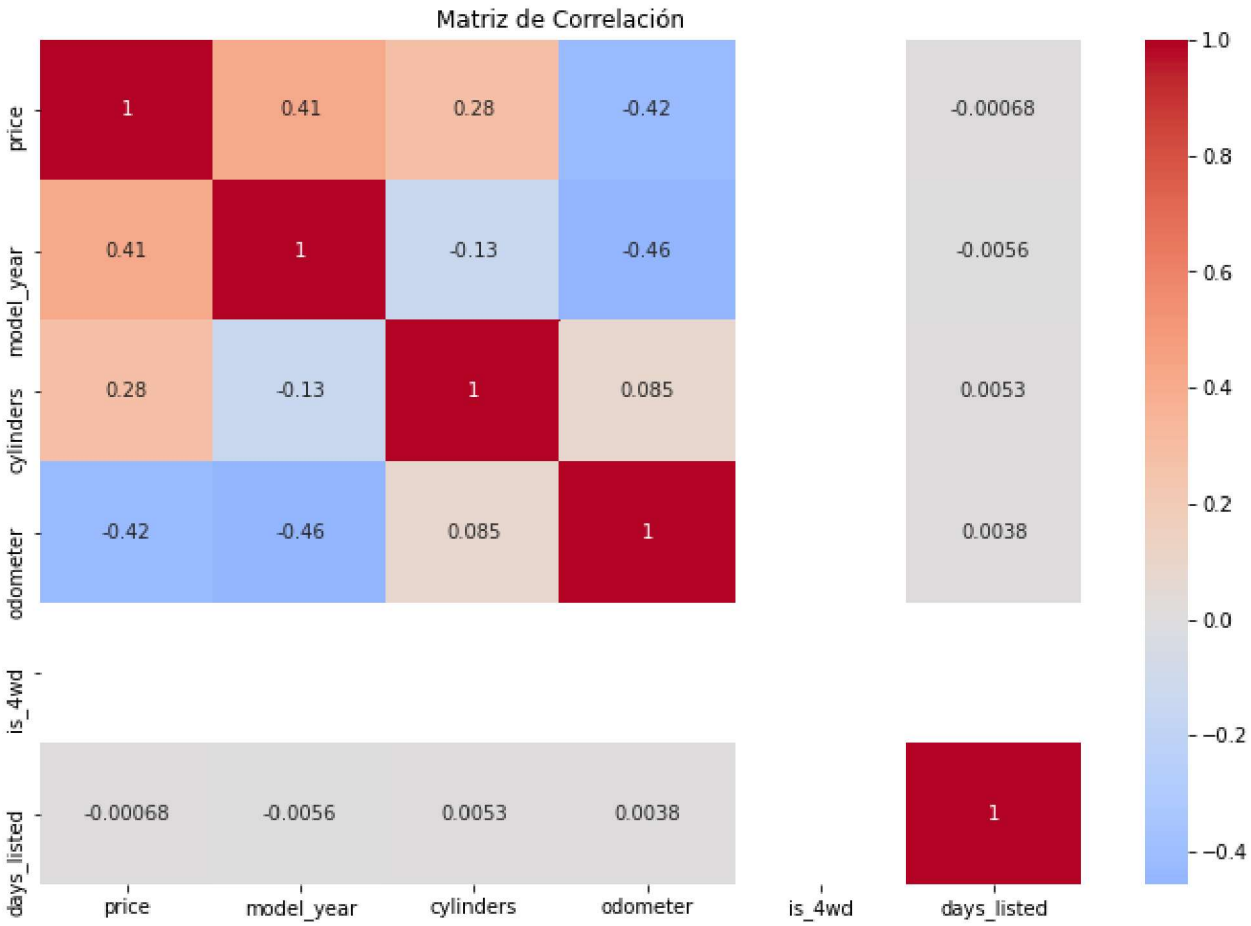
```
Cantidad de valores NaN en odometer: 7892
Media de odometer: 115553.4617376756
Mediana de odometer: 113000.0
```

**Respuesta del estudiante** En el caso de la columna $odometer$ es un poco mas complicado ya que la cantidad de valores nulos es un $15.31\ \%$ y esa cantidad de valores ya si entra en un

rango que podria afectar los resultados de manera negativa si no se trata adecuadamente.

In [10]:

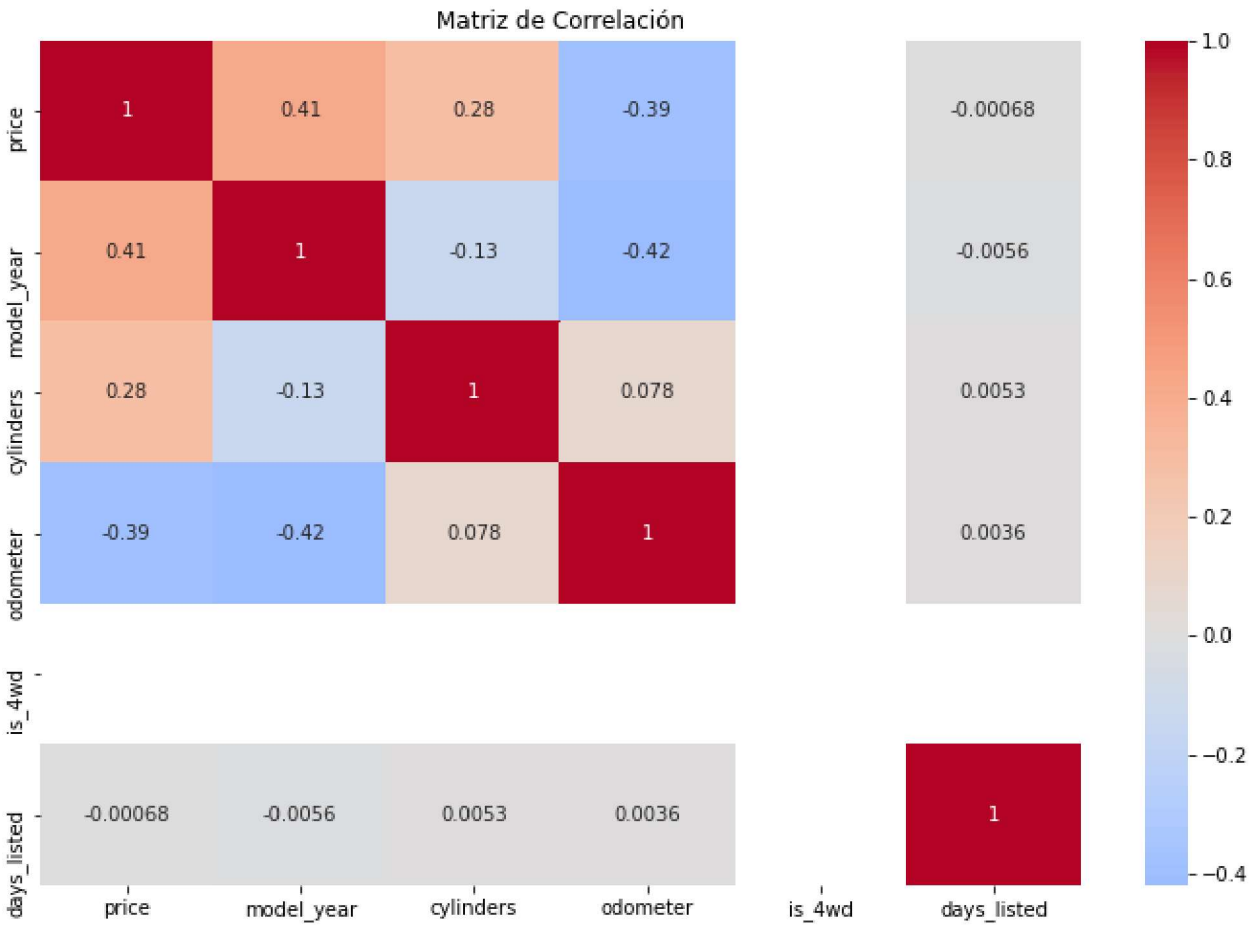
```
# Calcula la matriz de correlación

correlation_matrix = df.corr()

# Crea un mapa de calor para visualizar la matriz de correlación
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Matriz de Correlación')
plt.show()
```




**Respuesta del estudiante** Como se puede observar en la matriz de correlacion $odometer$ tiene una correlacion ligeramente negativa con $price$ y $\mod el_year$, lo que nos indica algo que es entendible, mientras mas alto el precio y mas nuevo el año, menor sera el millaje del vehiculo, podriamos decir que si estos 2 valores son cambiados bruscamente habria una incidencia negativa de mas de un 40% en la columna $odometer$, pero como los valores incluidos en estas columnas eran lo suficientemente limpios, no habra mucho problema, ahora podriamos intentar el reemplazo de estos valores en $odometer$ y ver que tanto afectaria nuestro grafico de correlacion, si no varia mucho, podriamos decir que logramos un reemplazo de valores nulos exitoso, ya que teniamos calculada la media y la mediana en esta columna previamente, vamos a sustituir los valores, esta vez, usaremos la mediana, ya que es la que menos varia los valores en general, y el resultado es menos problematico.

In [11]:
```
# eliminar los valores ausentes de 'odometer'

df['odometer'] = df['odometer'].fillna(median_odometer)
```

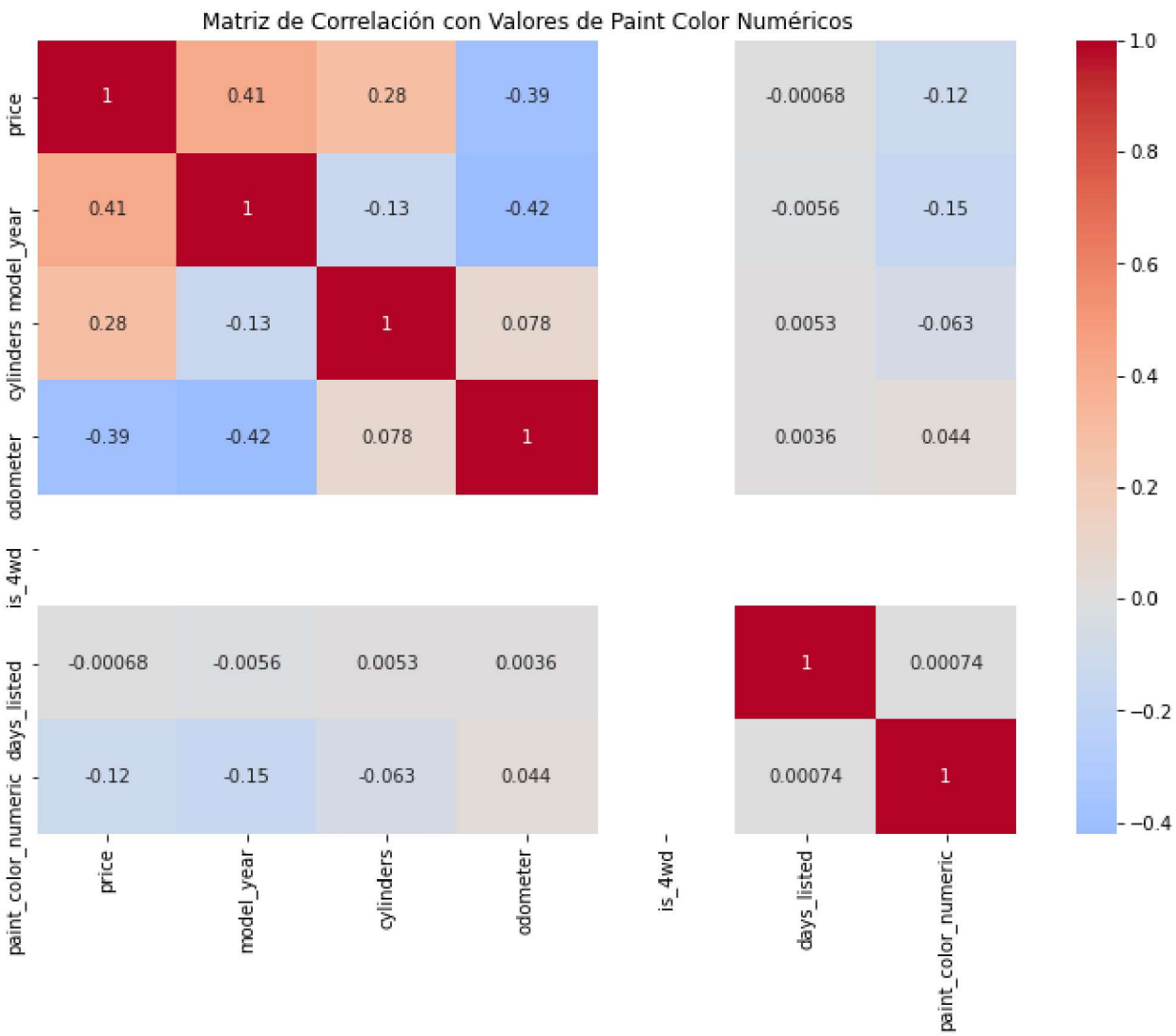
In [12]:
```
# Calcula la matriz de correlación

correlation_matrix = df.corr()

# Crea un mapa de calor para visualizar la matriz de correlación
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Matriz de Correlación')
plt.show()
```




**Respuesta del estudiante** Como podemos observar, haber cambiado los valores nulos por la mediana no tuvo una incidencia tan grande, que era lo que esperabamos, asi que estos valores los dejaremos asi, para poder continuar con la investigacion, si mas adelante apreciamos algun problema podriamos volver atras para corregir la resolucion de este problema.

In [13]:
```
# Crea una tabla de referencia para asignar valores numéricos a los colores

color_mapping = {
    'white': 1,
    'black': 2,
```

```
    'silver': 3,
    'grey': 4,
    'blue': 5,
    'red': 6,
    'green': 7,
    'brown': 8,
    'yellow': 9,
    'orange': 10,
    'purple': 11,
    'custom': 12
}

# Aplica el mapeo a la columna 'paint_color' para obtener valores numéricos
df['paint_color_numeric'] = df['paint_color'].map(color_mapping)

# Calcula la matriz de correlación entre la columna numérica 'paint_color_numeric' y ot
correlation_matrix = df.corr()

# Visualiza la matriz de correlación utilizando un mapa de calor
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Matriz de Correlación con Valores de Paint Color Numéricos')
plt.show()
```



**Respuesta del estudiante** Para tratar la columna $pa\int$, utilizaremos el color mas relevante ya que como variable categorica no hay ninguna relacion entre valores numericos y categoricos aparte de que pudimos aprecia que no hay una correlacion real con las demas columnas del dataframe, en el grafico anterior, podemos despreciar la nueva columna creada y trabajar directamente con la que ya tenemos rellenando los valores nulos con el color $white$ que es el mas abundante en el dataframe, ya que no afectara nuestra investigacion.

In [14]:

```
# Calcula la cantidad de veces que se repite cada color

color_counts = df['paint_color'].value_counts()

# Imprime la cantidad de veces que se repite cada color
color_counts
```

Out[14]:

```
white     10029
black      7692
silver     6244
grey       5037
blue       4475
red        4421
green      1396
brown      1223
custom     1153
yellow      255
orange      231
purple      102
Name: paint_color, dtype: int64
```

In [15]:

```
# Elimina la columna 'paint_color_numeric' si existe

if 'paint_color_numeric' in df.columns:
    df.drop('paint_color_numeric', axis=1, inplace=True)

# Filtra los colores que se repiten menos de 50 veces
relevant_colors = color_counts[color_counts >= 50].index

# Rellena los valores nulos en 'paint_color' con el color más común
most_common_color = color_counts.index[0]
df['paint_color'].fillna(most_common_color, inplace=True)

# Imprime el DataFrame con los valores nulos rellenados
df.head(10)
```

Out[15]:

| | price | model_year | model | condition | cylinders | fuel | odometer | transmission | type | paint_color |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 9400 | 2011.0 | bmw x5 | good | 6.0 | gas | 145000.0 | automatic | SUV | white |
| **1** | 25500 | 2011.0 | ford f-150 | good | 6.0 | gas | 88705.0 | automatic | pickup | white |
| **2** | 5500 | 2013.0 | hyundai sonata | like new | 4.0 | gas | 110000.0 | automatic | sedan | red |
| **3** | 1500 | 2003.0 | ford f-150 | fair | 8.0 | gas | 113000.0 | automatic | pickup | white |

| | price | model_year | model | condition | cylinders | fuel | odometer | transmission | type | paint_color |
|---|---|---|---|---|---|---|---|---|---|---|
| **4** | 14900 | 2017.0 | chrysler 200 | excellent | 4.0 | gas | 80903.0 | automatic | sedan | black |
| **5** | 14990 | 2014.0 | chrysler 300 | excellent | 6.0 | gas | 57954.0 | automatic | sedan | black |
| **6** | 12990 | 2015.0 | toyota camry | excellent | 4.0 | gas | 79212.0 | automatic | sedan | white |
| **7** | 15990 | 2013.0 | honda pilot | excellent | 6.0 | gas | 109473.0 | automatic | SUV | black |
| **8** | 11500 | 2012.0 | kia sorento | excellent | 4.0 | gas | 104174.0 | automatic | SUV | white |
| **9** | 9200 | 2008.0 | honda pilot | excellent | 6.0 | gas | 147191.0 | automatic | SUV | blue |

**Respuesta del estudiante** Como podemos apreciar los valores nulos faltantes fueron sustituidos, asi que ya tenemos un dataframe libre de valores nulos, ahora si podremos evaluar adecuadamente la data.

In [16]:

```
# Constancia de que los valores nulos fueron abordados correctamente

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51525 entries, 0 to 51524
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   price         51525 non-null  int64  
 1   model_year    51525 non-null  float64
 2   model         51525 non-null  object 
 3   condition     51525 non-null  object 
 4   cylinders     51525 non-null  float64
 5   fuel          51525 non-null  object 
 6   odometer      51525 non-null  float64
 7   transmission  51525 non-null  object 
 8   type          51525 non-null  object 
 9   paint_color   51525 non-null  object 
 10  is_4wd        25572 non-null  float64
 11  date_posted   51525 non-null  object 
 12  days_listed   51525 non-null  int64  
dtypes: float64(4), int64(2), object(7)
memory usage: 5.1+ MB
```

# Corregir los tipos de datos

**Respuesta del estudiante** Vamos a corregir en primera instancia el tipo de variable usado por $\mathrm{mod}\ el_year$ (deberia usar valores enteros como representacion de los años) y luego el tipo de variable para $is_4wd$ (deberia usar una variable bool, para simplificar su entendimiento), por ultimo corregiremos $date_posted$, ya que esta usando una variable tipo object, lo cual no nos

permitira trabajar con ella de manera normal (como una fecha real), entonces despues de cambiar el tipo de variable a tipo datetime, si podriamos usar la fecha como una fecha real, en esta ocasion hemos estado discriminando la columna $pa\int$ porque necesita un tratamiento un tanto especial.

In [17]:

```
# cambiar tipo de variables a las variables correctas, que permitiran mejor evaluacion

df['model_year'] = df['model_year'].astype('int64')
df['odometer'] = df['odometer'].astype('int64')
df['is_4wd'] = df['is_4wd'].fillna(False).astype(bool)
```

# Enriquecer datos

In [18]:

```
# Agrega valores de fecha y hora para cuando se colocó el anuncio

df['date_posted'] = pd.to_datetime(df['date_posted'])
```

**Respuesta del estudiante** Como se puede observar los valores de $date_posted$ no tienen hora, pero ya que no tenemos esa informacion y no se utilizara en la investigacion, es irrelevante, podemos continuar sin problemas.

In [19]:

```
# Agrega los años del vehículo cuando el anuncio se colocó

df['vehicle_age'] = df['date_posted'].dt.year - df['model_year']
```

**Respuesta del estudiante** Este es un dato interesante ya que con el podremos saber la edad real del vehiculo que el cliente puede adquirir.

In [20]:

```
# Agrupa los datos por año del modelo y calcula el millaje promedio por año

average_mileage_by_year = df.groupby('model_year')['odometer'].mean()

# Convierte la serie de millaje promedio de float64 a int64
average_mileage_by_year = average_mileage_by_year.astype('int64')

# Agrega la columna de millaje promedio por año al DataFrame
df['average_mileage_by_year'] = df['model_year'].map(average_mileage_by_year)
```

**Respuesta del estudiante** La informacion del millaje promedio es super valiosa, ya que asi podremos saber que tanto abusaron del vehiculo antes de ponerlo en venta, eso tambien nos ayuda a saber que vehiculo tiene menor desgaste y esto podria llevar a un precio final mayor.

In [21]:

```
# Podría ser útil reemplazar los valores de condición con algo que se pueda manipular m
```

```
# Define el mapeo de valores de cadena a valores numéricos
condition_mapping = {
    'new': 5,
    'like new': 4,
    'excellent': 3,
    'good': 2,
    'fair': 1,
    'salvage': 0
}

# Reemplaza los valores de la columna 'condition' con los valores numéricos
df['condition'] = df['condition'].replace(condition_mapping)
```

**Respuesta del estudiante** Al cambiar los valores en *condition*, por estos numericos nos simplificaremos la evaluacion, ya que los valores numericos son mas faciles de evaluar con miras al estudio de las correlaciones entre columna y demas.

# Comprobar datos limpios

In [22]:

```
# imprime la información general/resumida sobre el DataFrame

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51525 entries, 0 to 51524
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   price                    51525 non-null  int64
 1   model_year               51525 non-null  int64
 2   model                    51525 non-null  object
 3   condition                51525 non-null  int64
 4   cylinders                51525 non-null  float64
 5   fuel                     51525 non-null  object
 6   odometer                 51525 non-null  int64
 7   transmission             51525 non-null  object
 8   type                     51525 non-null  object
 9   paint_color              51525 non-null  object
 10  is_4wd                   51525 non-null  bool
 11  date_posted              51525 non-null  datetime64[ns]
 12  days_listed              51525 non-null  int64
 13  vehicle_age              51525 non-null  int64
 14  average_mileage_by_year  51525 non-null  int64
dtypes: bool(1), datetime64[ns](1), float64(1), int64(7), object(5)
memory usage: 5.6+ MB
```

**Respuesta del estudiante** Como podemos observar ya los datos estan limpios y listos para ser utilizados en la investigacion, todas las columnas tienen la misma cantidad de filas, sin datos ausentes ya podemos proceder a abordar la investigacion en torno a los datos atipicos, para poder llegar a unas conclusiones mas certeras.

In [23]:

```
# imprimir una muestra de datos

df.head(10)
```

Out[23]:

| | price | model_year | model | condition | cylinders | fuel | odometer | transmission | type | paint_color |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 9400 | 2011 | bmw x5 | 2 | 6.0 | gas | 145000 | automatic | SUV | white |
| **1** | 25500 | 2011 | ford f-150 | 2 | 6.0 | gas | 88705 | automatic | pickup | white |
| **2** | 5500 | 2013 | hyundai sonata | 4 | 4.0 | gas | 110000 | automatic | sedan | red |
| **3** | 1500 | 2003 | ford f-150 | 1 | 8.0 | gas | 113000 | automatic | pickup | white |
| **4** | 14900 | 2017 | chrysler 200 | 3 | 4.0 | gas | 80903 | automatic | sedan | black |
| **5** | 14990 | 2014 | chrysler 300 | 3 | 6.0 | gas | 57954 | automatic | sedan | black |
| **6** | 12990 | 2015 | toyota camry | 3 | 4.0 | gas | 79212 | automatic | sedan | white |
| **7** | 15990 | 2013 | honda pilot | 3 | 6.0 | gas | 109473 | automatic | SUV | black |
| **8** | 11500 | 2012 | kia sorento | 3 | 4.0 | gas | 104174 | automatic | SUV | white |
| **9** | 9200 | 2008 | honda pilot | 3 | 6.0 | gas | 147191 | automatic | SUV | blue |

# Estudiar parámetros principales

Los parámetros son:

- Precio
- Los años del vehículo cuando el anuncio se colocó
- Millaje
- Número de cilindros
- Estado

**Respuesta del estudiante** En esta seccion vamos a abordar los datos atipicos, visualizaremos las columnas que tienen mayor relevancia en cuanto a lo que nos interesa de la investigacion, y basados en los histogramas que obtengamos de ellas entonces discriminaremos los valores atipicos.

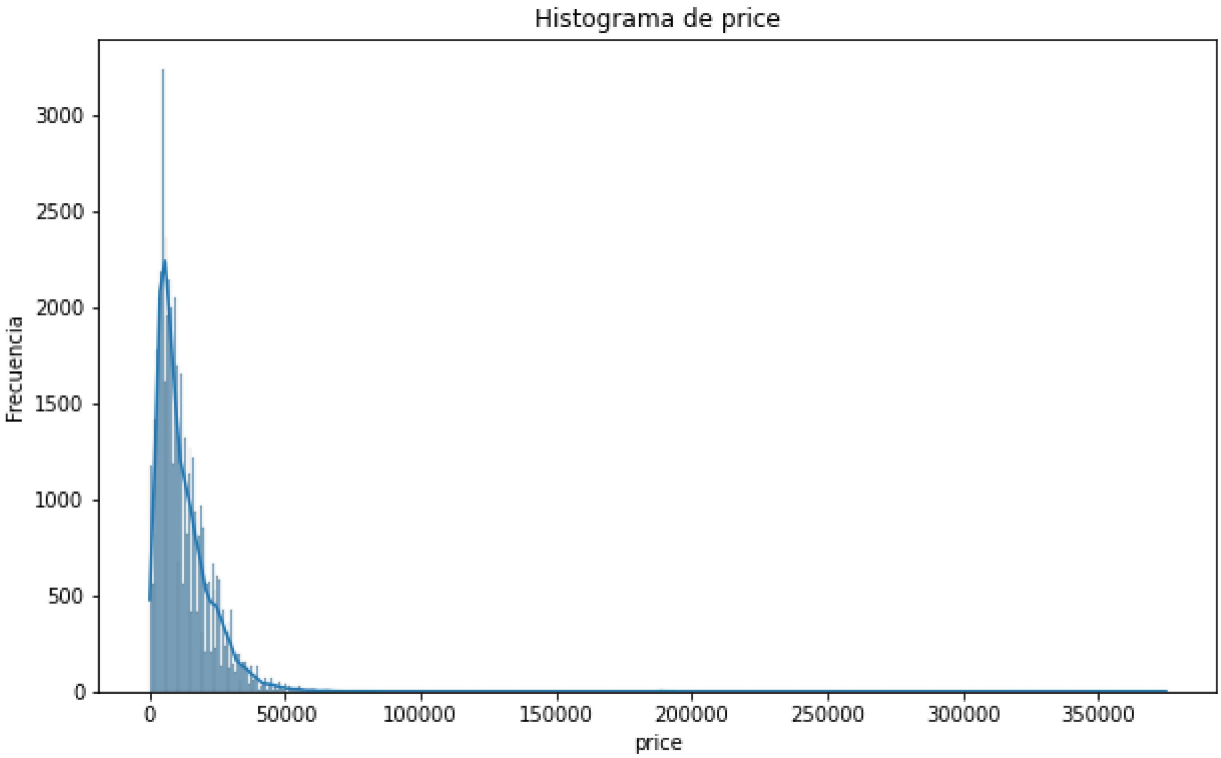
In [24]:

```
# Función para generar histograma con análisis de valores atípicos

def analyze_parameter(column_name):
    plt.figure(figsize=(10, 6))
    sns.histplot(data=df, x=column_name, kde=True)
    plt.title(f'Histograma de {column_name}')
    plt.xlabel(column_name)
```

```
    plt.ylabel('Frecuencia')
    plt.show()
```

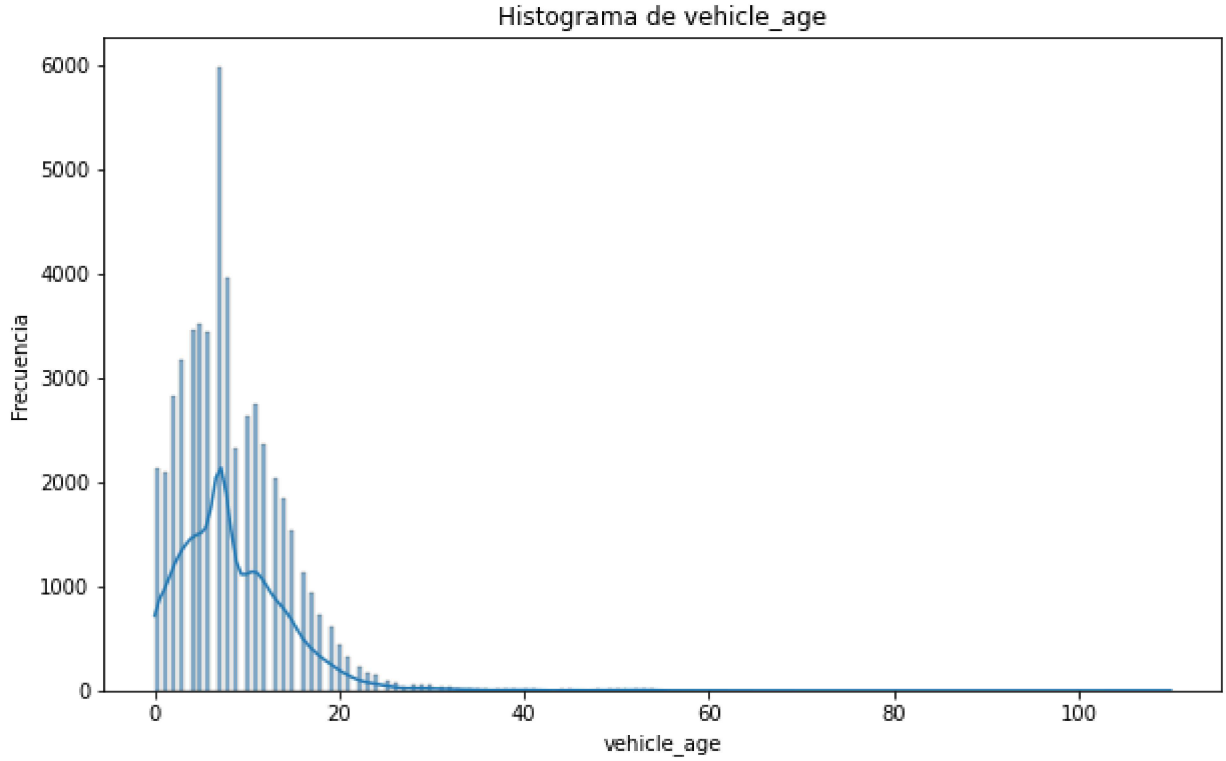
In [25]:

```
# Llamada a la función para analizar la columna 'price'

analyze_parameter('price')
```



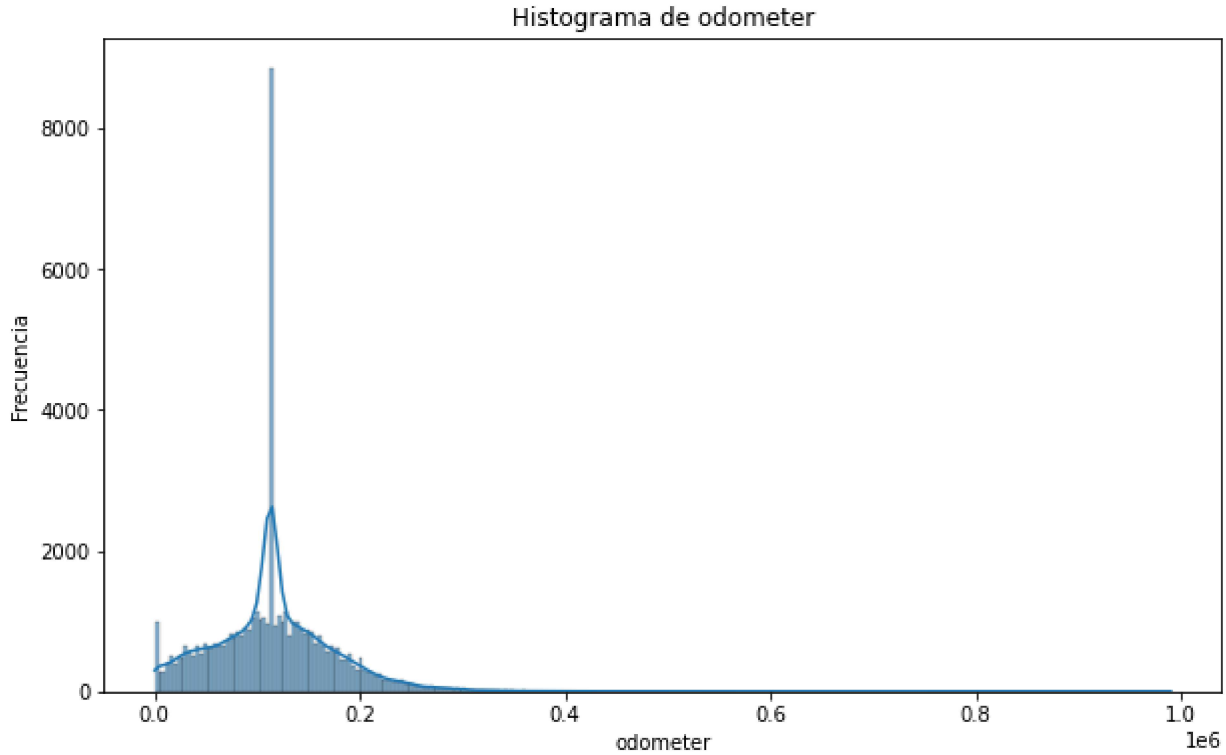

In [26]:

```
# Llamada a la función para analizar la columna 'vehicle_age'

analyze_parameter('vehicle_age')
```



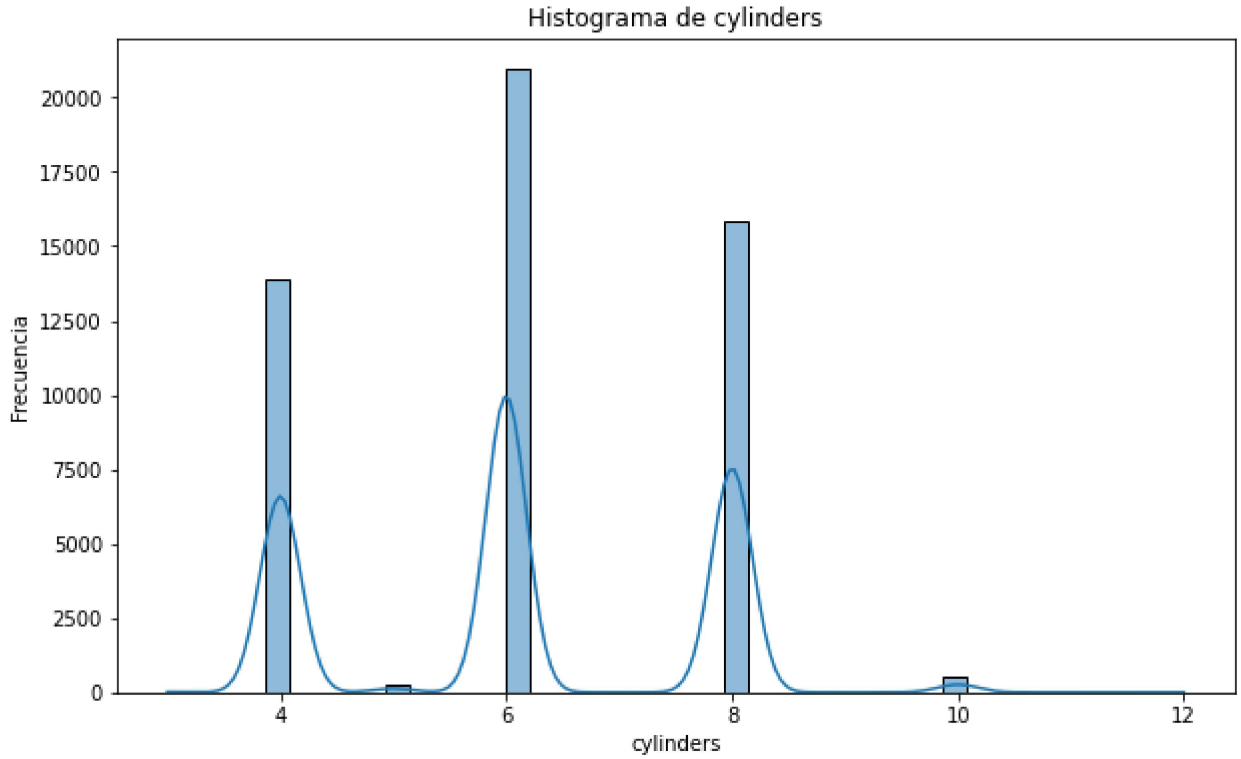

In [27]:

```
# Llamada a la función para analizar la columna 'odometer'

analyze_parameter('odometer')
```



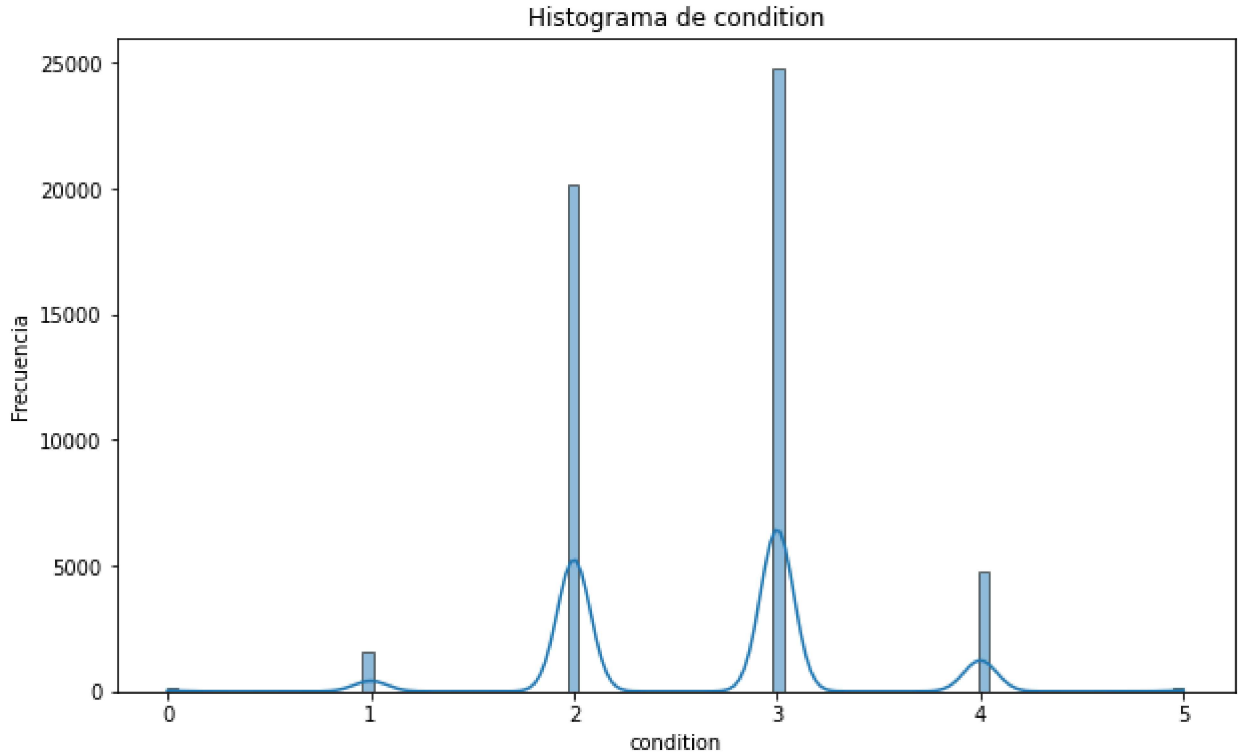

In [28]:

```
# Llamada a la función para analizar la columna 'cylinders'

analyze_parameter('cylinders')
```




In [29]:

```
# Llamada a la función para analizar la columna 'condition'

analyze_parameter('condition')
```




**Respuesta del estudiante** Como podemos observar en los graficos, cerca del $10\,\%$ de la data tiene una tendencia atipica, esto lo veremos en la parte siguiente donde determinaremos cuales

valores seran nuestro target como valores atipicos, para discriminarlos de forma efectiva y poder seguir adelante con la investigacion.

# Estudiar y tratar valores atípicos

In [30]:

```
# Determina los límites inferiores para valores atípicos

def calculate_lower_bound(column):
    q1 = column.quantile(0.25)
    q3 = column.quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    return lower_bound

lower_bound_price = calculate_lower_bound(df['price'])
lower_bound_vehicle_age = calculate_lower_bound(df['vehicle_age'])
lower_bound_odometer = calculate_lower_bound(df['odometer'])
lower_bound_cylinders = calculate_lower_bound(df['cylinders'])
lower_bound_condition = calculate_lower_bound(df['condition'])

print("Límites inferiores para valores atípicos:")
print("Price:", lower_bound_price)
print("Vehicle Age:", lower_bound_vehicle_age)
print("Odometer:", lower_bound_odometer)
print("Cylinders:", lower_bound_cylinders)
print("Condition:", lower_bound_condition)
```

```
Límites inferiores para valores atípicos:
Price: -12758.5
Vehicle Age: -8.0
Odometer: -21859.0
Cylinders: -2.0
Condition: 0.5
```

**Respuesta del estudiante** Como podemos observar que hay valores atipicos bien marcados en los limites inferiores de la columna $price$ y $odometer$, aparte tambien las columnas $vehic \leq_a \geq$ y $cyl \in ders$ tienen algunos de esos valores incoherentes, al igual que $condition$ aunque esta ultima en menor medida.

In [31]:

```
# Determinar los límites superiores para valores atípicos

def calculate_upper_bound(column):
    q1 = column.quantile(0.25)
    q3 = column.quantile(0.75)
    iqr = q3 - q1
    upper_bound = q3 + 1.5 * iqr
    return upper_bound

upper_bound_price = calculate_upper_bound(df['price'])
upper_bound_vehicle_age = calculate_upper_bound(df['vehicle_age'])
upper_bound_odometer = calculate_upper_bound(df['odometer'])
upper_bound_cylinders = calculate_upper_bound(df['cylinders'])
upper_bound_condition = calculate_upper_bound(df['condition'])
```

```
print("Límites superiores para valores atípicos:")
print("Price:", upper_bound_price)
print("Vehicle Age:", upper_bound_vehicle_age)
print("Odometer:", upper_bound_odometer)
print("Cylinders:", upper_bound_cylinders)
print("Condition:", upper_bound_condition)
```

```
Límites superiores para valores atípicos:
Price: 34597.5
Vehicle Age: 24.0
Odometer: 247581.0
Cylinders: 14.0
Condition: 4.5
```

**Respuesta del estudiante** Igual por aca, podemos observar que hay valores atipicos en los limites superiores de la columna $price$ y $odometer$, aparte tambien las columnas $vehic \leq_a \geq$ y $cyl \in ders$ tienen algunos de esos valores atipicos, al igual que $condition$ aunque esta ultima en menor medida, igual que la anterior vez.

In [32]:

```
# Almacena los datos sin valores atípicos en un DataFrame separado

filtered_df = df[
    (df['price'] >= 1000) & (df['price'] <= 34597) &
    (df['vehicle_age'] >= 0) & (df['vehicle_age'] <= 24) &
    (df['odometer'] >= 0) & (df['odometer'] <= 247581) &
    (df['cylinders'] >= 2) & (df['cylinders'] <= 12) &
    (df['condition'] >= 0) & (df['condition'] <= 5)
]

# Mostrar información general del nuevo DataFrame sin valores atípicos
print("Información general del DataFrame sin valores atípicos:")
filtered_df.info()
```

```
Información general del DataFrame sin valores atípicos:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 46727 entries, 0 to 51524
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   price                   46727 non-null  int64
 1   model_year              46727 non-null  int64
 2   model                   46727 non-null  object
 3   condition               46727 non-null  int64
 4   cylinders               46727 non-null  float64
 5   fuel                    46727 non-null  object
 6   odometer                46727 non-null  int64
 7   transmission            46727 non-null  object
 8   type                    46727 non-null  object
 9   paint_color             46727 non-null  object
 10  is_4wd                  46727 non-null  bool
 11  date_posted             46727 non-null  datetime64[ns]
 12  days_listed             46727 non-null  int64
 13  vehicle_age             46727 non-null  int64
 14  average_mileage_by_year 46727 non-null  int64
dtypes: bool(1), datetime64[ns](1), float64(1), int64(7), object(5)
memory usage: 5.4+ MB
```

**Respuesta del estudiante** Aqui ya hemos procedido a mitigar el inconveniente de los valores atipicos, nos hemos quedado con el $90.68\ \%$ de la data del dataframe original para la

investigacion, lo que indica que la discriminacion de estos valores no va a influir de manera negativa con el trabajo que debe realizarse.

In [33]:

```
# Mostrar las primeras filas del DataFrame filtrado

filtered_df.head(10)
```

Out[33]:

| | price | model_year | model | condition | cylinders | fuel | odometer | transmission | type | paint_color |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 9400 | 2011 | bmw x5 | 2 | 6.0 | gas | 145000 | automatic | SUV | white |
| **1** | 25500 | 2011 | ford f-150 | 2 | 6.0 | gas | 88705 | automatic | pickup | white |
| **2** | 5500 | 2013 | hyundai sonata | 4 | 4.0 | gas | 110000 | automatic | sedan | red |
| **3** | 1500 | 2003 | ford f-150 | 1 | 8.0 | gas | 113000 | automatic | pickup | white |
| **4** | 14900 | 2017 | chrysler 200 | 3 | 4.0 | gas | 80903 | automatic | sedan | black |
| **5** | 14990 | 2014 | chrysler 300 | 3 | 6.0 | gas | 57954 | automatic | sedan | black |
| **6** | 12990 | 2015 | toyota camry | 3 | 4.0 | gas | 79212 | automatic | sedan | white |
| **7** | 15990 | 2013 | honda pilot | 3 | 6.0 | gas | 109473 | automatic | SUV | black |
| **8** | 11500 | 2012 | kia sorento | 3 | 4.0 | gas | 104174 | automatic | SUV | white |
| **9** | 9200 | 2008 | honda pilot | 3 | 6.0 | gas | 147191 | automatic | SUV | blue |

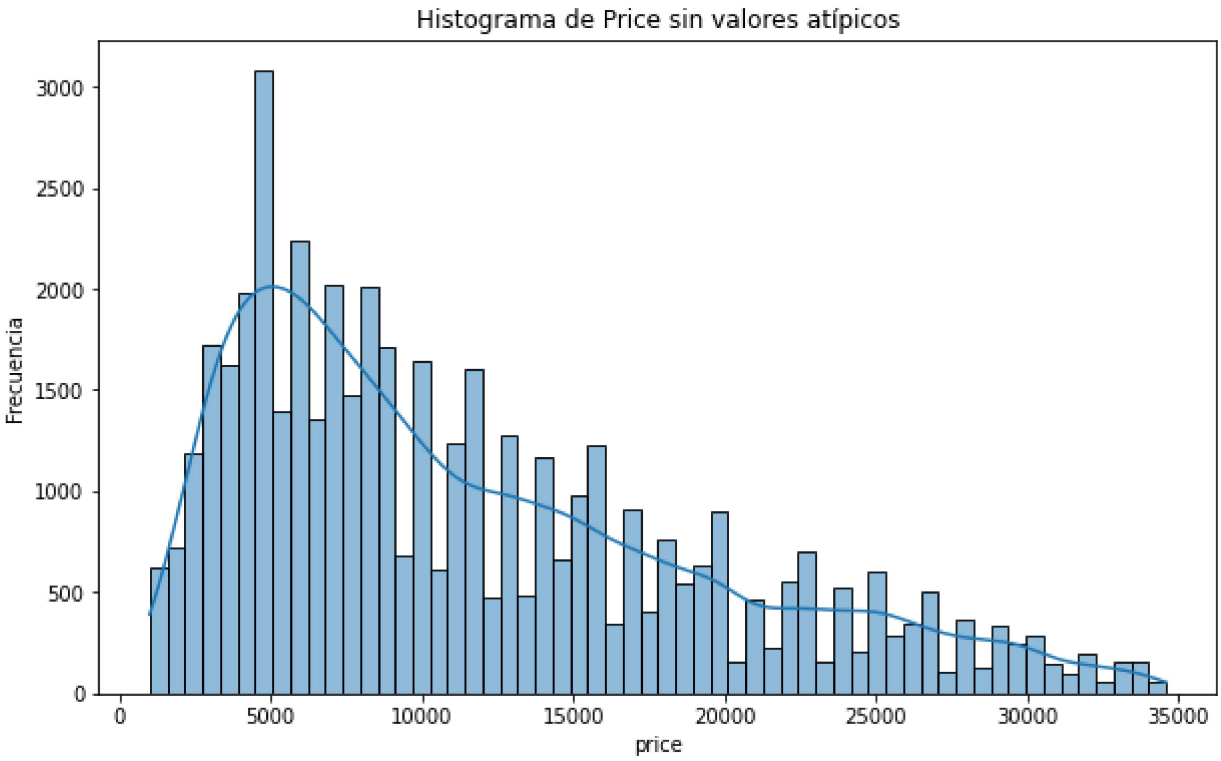
**Respuesta del estudiante** Aca podemos apreciar una muestra del dataframe filtrado, despues de la discriminacion de los valores atipicos.

# Estudiar parámetros principales sin valores atípicos

In [34]:

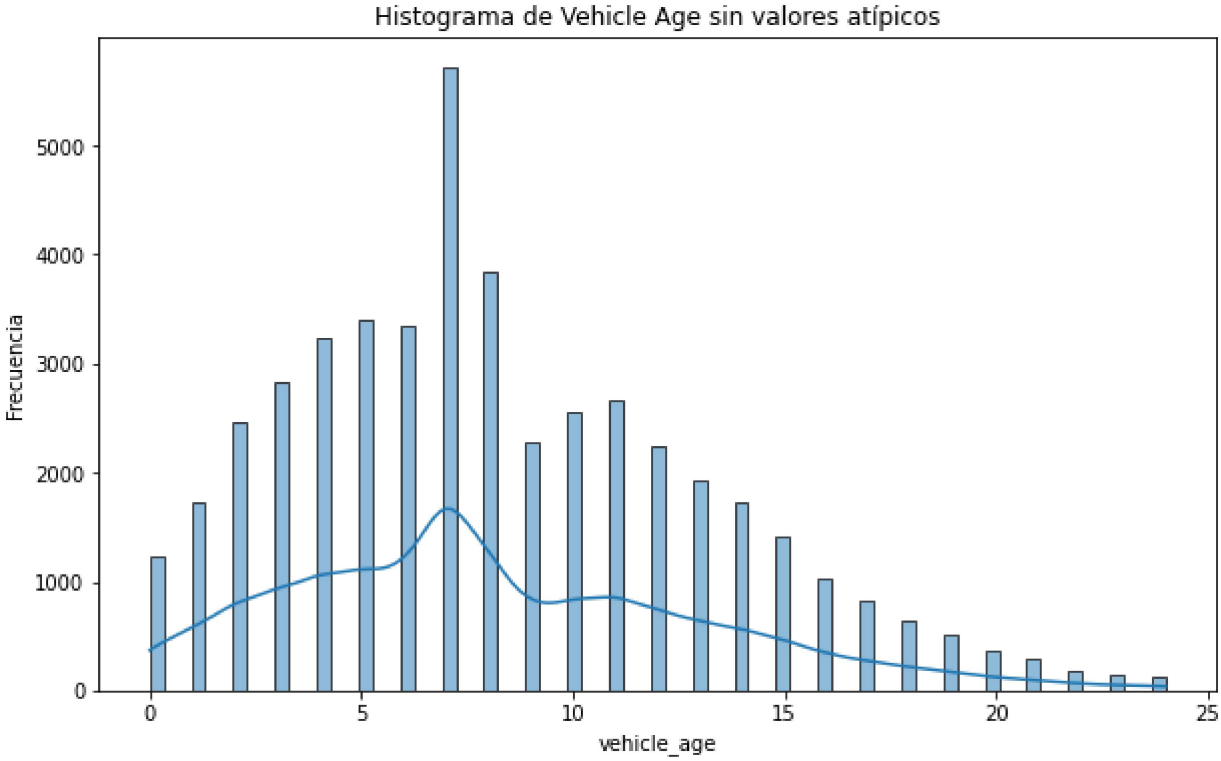
```
# Definir función para crear histogramas

def create_histogram(column_name, title):
    plt.figure(figsize=(10, 6))
    sns.histplot(data=filtered_df, x=column_name, kde=True)
    plt.title(title)
    plt.xlabel(column_name)
    plt.ylabel('Frecuencia')
    plt.show()
```

In [35]:
```
# Crear histograma para 'price'

create_histogram('price', 'Histograma de Price sin valores atípicos')
```




**Respuesta del estudiante** Comparando el nuevo histograma de $price$, la data parece tener un mayor nivel de relevancia, ya que es mas entendible y amigable para los no entendidos, se aprecia la misma tendencia a tener mas publicaciones con precios alrededor de los 5000, lo cual nos permite saber que acertamos al discriminar los valores que sacamos del dataframe sin filtrar.

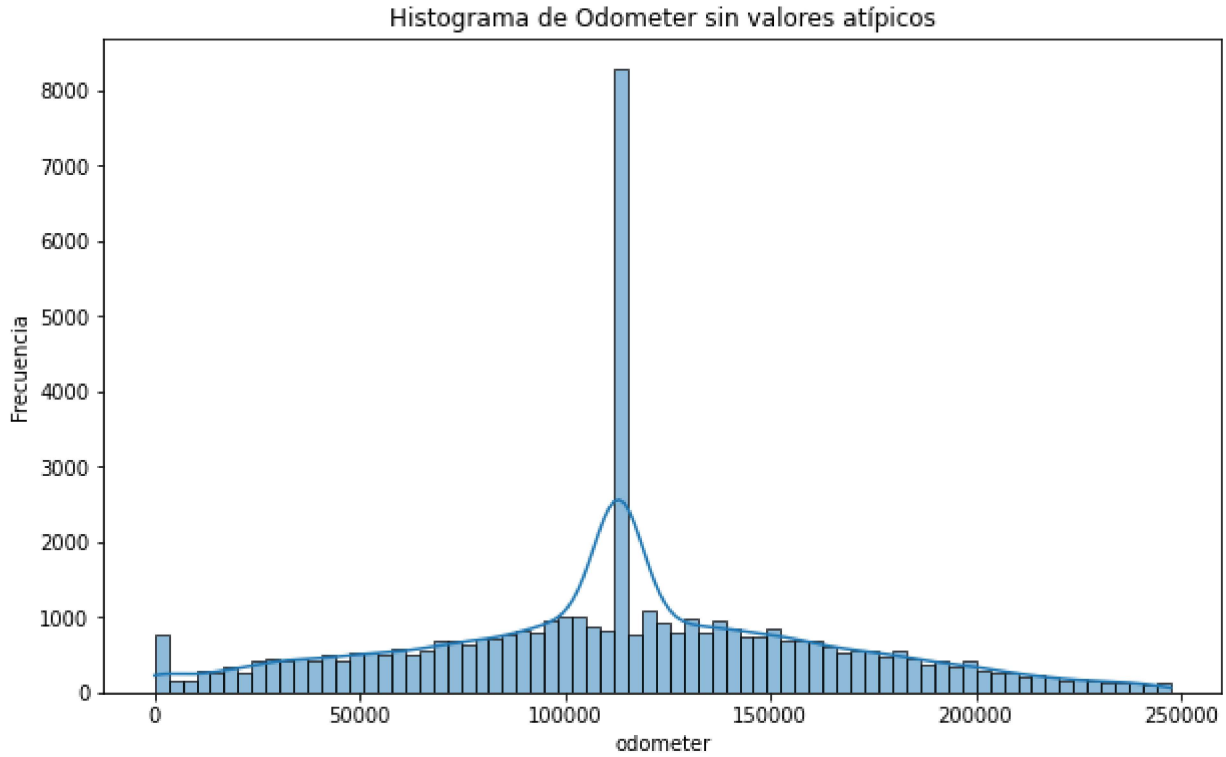
In [36]:
```
# Crear histograma para 'vehicle_age'

create_histogram('vehicle_age', 'Histograma de Vehicle Age sin valores atípicos')
```




**Respuesta del estudiante** En este nuevo grafico de $vehic \leq_a \geq$ podemos observar la misma tendencia a publicaciones de vehiculos con edades que rondan los 7 años, lo cual es lo mismo que se observaba en el anterior grafico, pero la verdad era poco entendible, si no se hubiera hecho la discriminacion de valores atipicos, ese anterior grafico nos habria servido de poca cosa.

In [37]:

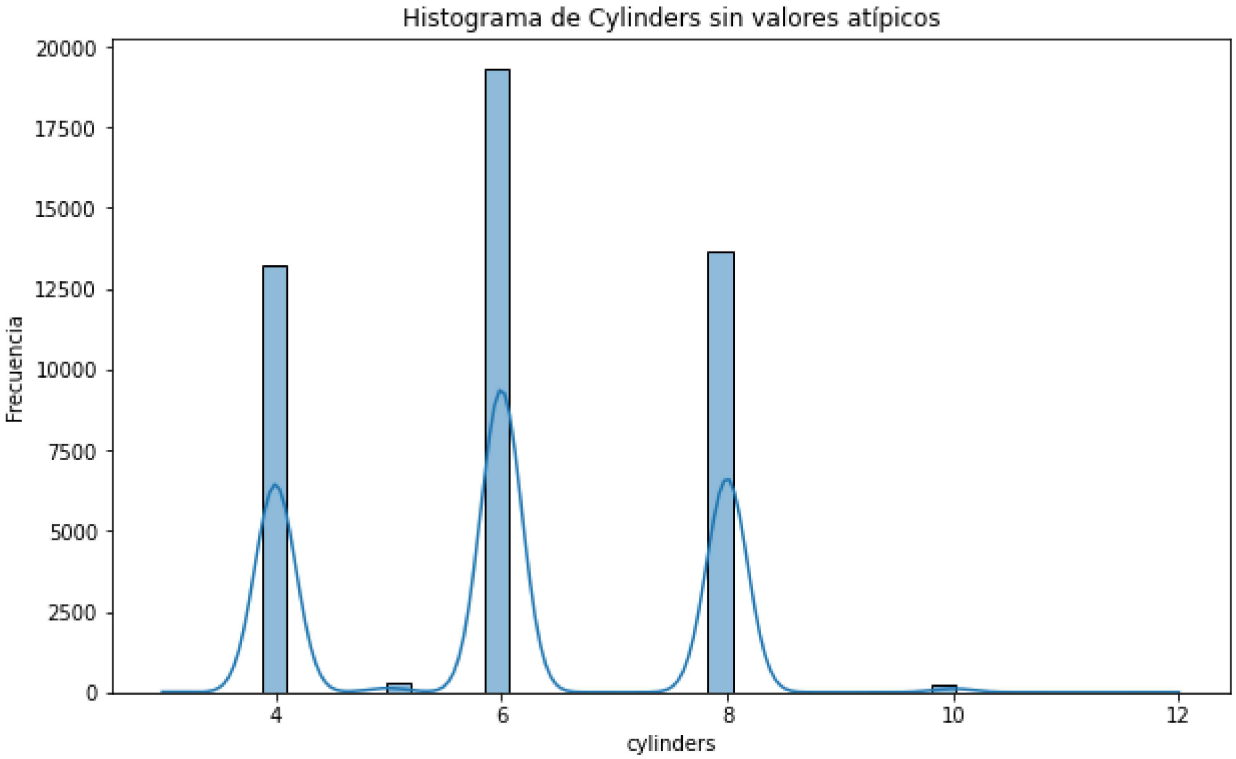
```
# Crear histograma para 'odometer'

create_histogram('odometer', 'Histograma de Odometer sin valores atípicos')
```




**Respuesta del estudiante** En esta nueva iteracion de la columna $odometer$, podemos apreciar tendencias iguales, con mejor visibilidad, mas entendibles, al igual que los demas graficos y con datos utiles y listos para una investigacion mas profunda, como podemos ver la tendencia sigue siendo a valores que rondan las $120000$ millas, lo cual nos permite tener en cuenta este tipo de valores a la hora de nuestras conclusiones preliminares.

In [38]:

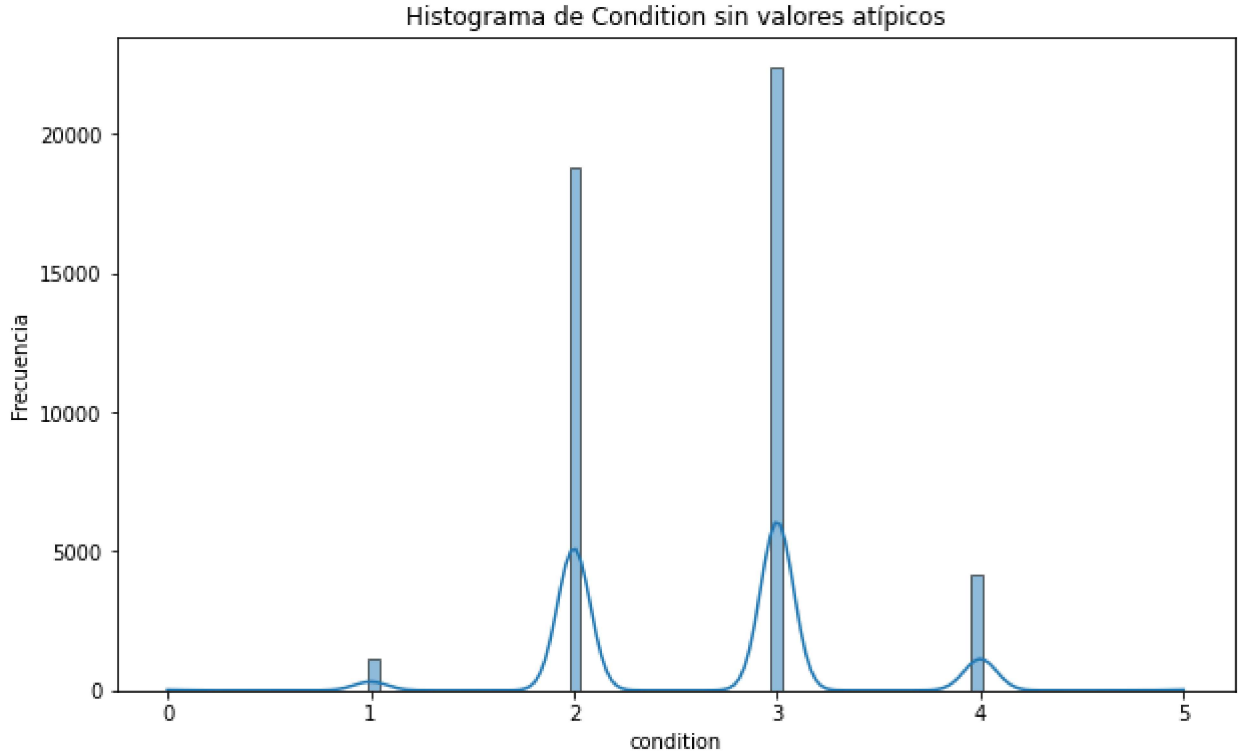
```
# Crear histograma para 'cylinders'

create_histogram('cylinders', 'Histograma de Cylinders sin valores atípicos')
```




**Respuesta del estudiante** El grafico de esta columna $cyl \in ders$, no tiene mucho de donde quitarle, asi que basicamente es el mismo que vimos con valores atipicos, pero sin los valores atipicos que tenia, podemos ver que la mayor cantidad de publicaciones que tienen una cilindrada especifica se centran en los $6$ cilindros.

In [39]:

```
# Crear histograma para 'condition'

create_histogram('condition', 'Histograma de Condition sin valores atípicos')
```



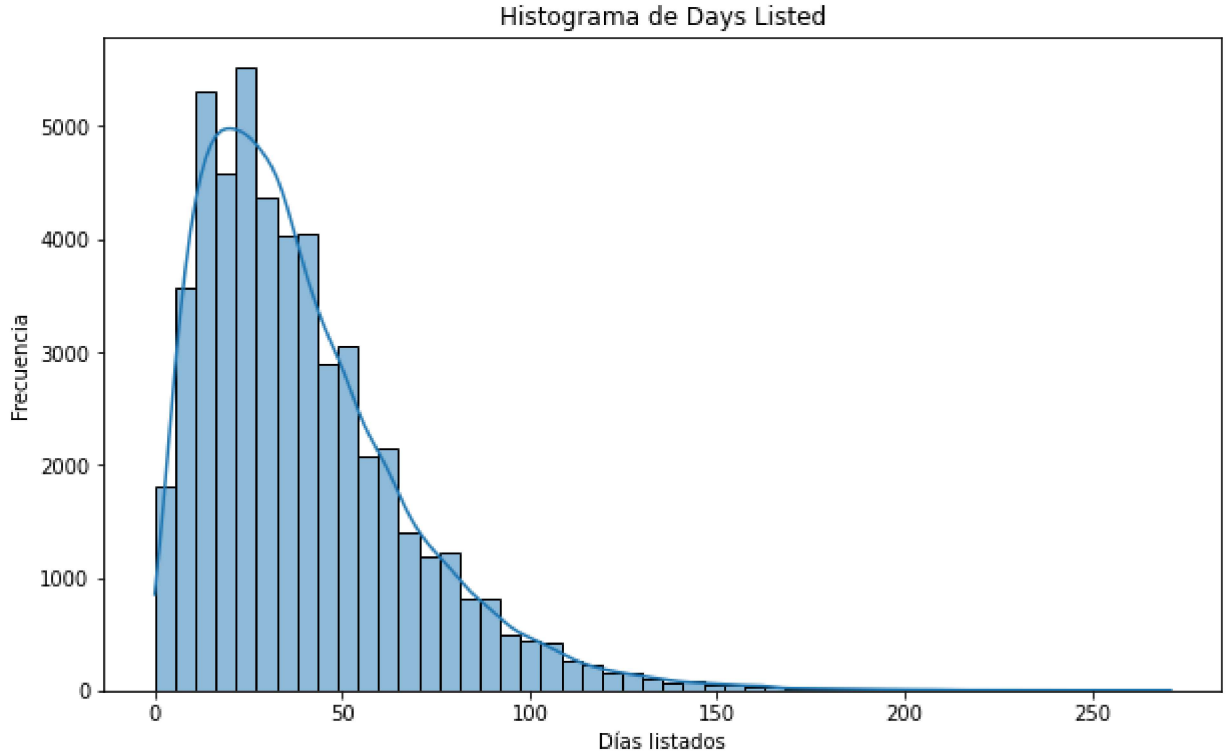

**Respuesta del estudiante** En este grafico de la columna $condition$ podemos ver la misma tendencia del anterior pero con la data ya limpia, la mayor cantidad de publicaciones se centran en vehiculos en condicion $3$ que seria $excel \leq nt$ y como segundo lugar estan los que estan en $g\infty d.$

# Periodo de colocación de los anuncios

In [40]:

```
# Calcular la media y la mediana de days_listed

mean_days_listed = df['days_listed'].mean()
median_days_listed = df['days_listed'].median()

print("Media de días listados:", mean_days_listed)
print("Mediana de días listados:", median_days_listed)

# Crear un histograma de days_listed
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='days_listed', bins=50, kde=True)
plt.title('Histograma de Days Listed')
plt.xlabel('Días listados')
plt.ylabel('Frecuencia')
plt.show()
```

```
Media de días listados: 39.55475982532751
Mediana de días listados: 33.0
```




**Respuesta del estudiante** Como podemos observar el tiempo habitual de colocacion de un anuncio ronda los $33$ a $39$ dias, segun la media y la mediana respectivamente.

In [41]:

```
# Definir umbrales para anuncios eliminados rápidamente y anuncios de duración anormalm

quick_removal_threshold = 10
long_duration_threshold = 60

# Filtrar anuncios eliminados rápidamente y con duración anormalmente larga
quick_removal_ads = df[df['days_listed'] < quick_removal_threshold]
long_duration_ads = df[df['days_listed'] > long_duration_threshold]

# Imprimir la cantidad de anuncios eliminados rápidamente y de duración anormalmente la
print("Cantidad de anuncios eliminados rápidamente:", len(quick_removal_ads))
print("Cantidad de anuncios de duración anormalmente larga:", len(long_duration_ads))
```

```
Cantidad de anuncios eliminados rápidamente: 4586
Cantidad de anuncios de duración anormalmente larga: 9891
```

In [42]:

```
# Obtener los modelos, edades y condiciones de vehículos que fueron eliminados rápidame

quick_removal_info = quick_removal_ads[['model', 'vehicle_age', 'condition']]

# Obtener los modelos, edades y condiciones de vehículos con duración anormalmente larg
long_duration_info = long_duration_ads[['model', 'vehicle_age', 'condition']]

# Mostrar los modelos, edades y condiciones con mayor incidencia de eliminación rápida
print("Modelos, edades y condiciones con mayor incidencia de eliminación rápida:")
print(quick_removal_info.head(10))

# Mostrar los modelos, edades y condiciones con mayor incidencia de duración anormalmen
```

```
print("Modelos, edades y condiciones con mayor incidencia de duración anormalmente larg
print(long_duration_info.head(10))
```

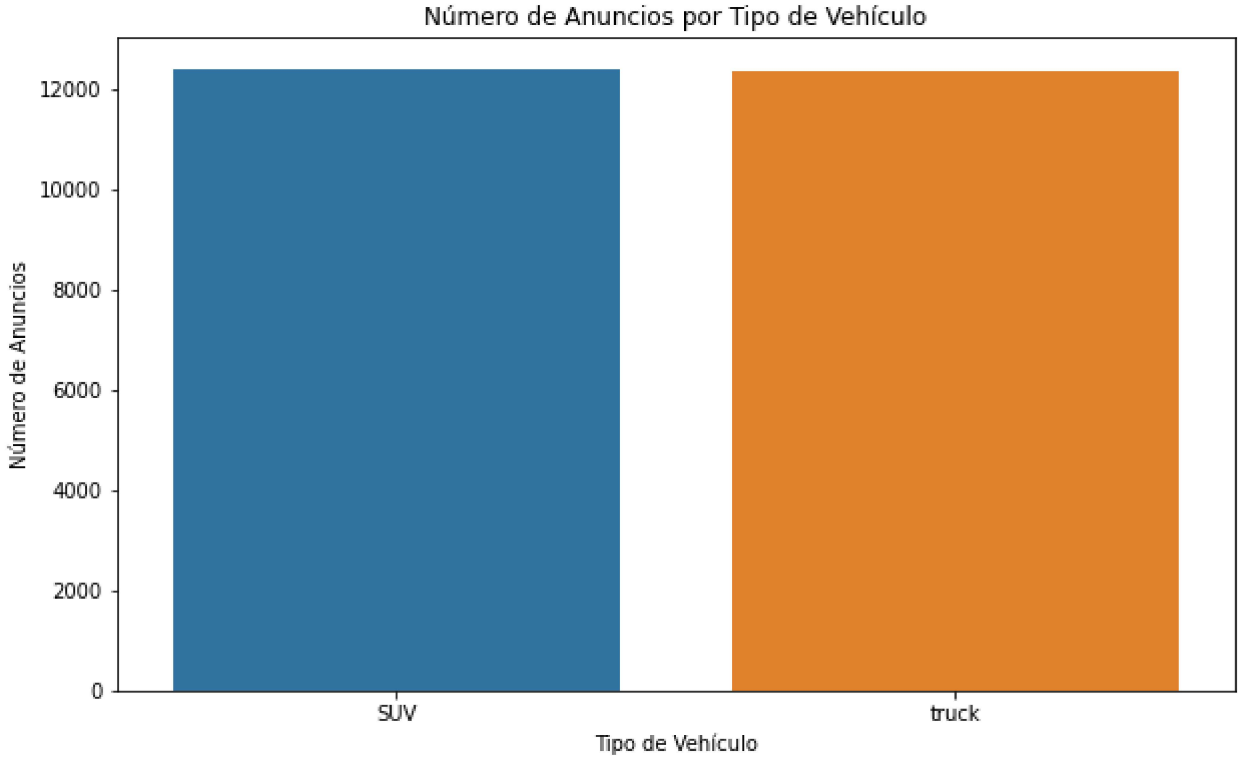
```
Modelos, edades y condiciones con mayor incidencia de eliminación rápida:
                            model  vehicle_age  condition
3                       ford f-150           16          1
21                     toyota rav4           11          2
32                   nissan altima            3          4
39                      ford f-150           10          2
59                toyota highlander           12          2
73      jeep grand cherokee laredo           24          2
75                 chevrolet tahoe           13          3
84                 hyundai elantra            7          4
114    nissan frontier crew cab sv            7          2
120  chevrolet silverado 1500 crew            1          2
Modelos, edades y condiciones con mayor incidencia de duración anormalmente larga:
                  model  vehicle_age  condition
2        hyundai sonata            6          4
6          toyota camry            3          3
7           honda pilot            6          3
15             ram 1500            5          3
22  chevrolet silverado           15          4
28   chevrolet corvette            6          3
34    toyota highlander           10          2
46        dodge charger            3          3
51           ford focus            6          2
52             kia soul            8          2
```

**Respuesta del estudiante** Aca podemos observar que los anuncios de los vehiculos que se eliminaron de manera mas rapida fueron vehiculos con poco tiempo de uso, y los que estuvieron publicados por mayor tiempo en la pagina web fueron vehiculos con mas de 6 años de uso y $fair$ en la columna $condition$, asi que podemos decir que los vehiculos que fueron mas abusados tuvieron peor salida sin mucha variacion en la cantidad de tiempo de uso.

# Precio promedio por cada tipo de vehículo

In [43]:

```
# Calcular el número de anuncios y el precio promedio por tipo de vehículo

vehicle_type_stats = df.groupby('type').agg({'price': 'mean', 'model': 'count'}).reset_
vehicle_type_stats.rename(columns={'model': 'num_ads'}, inplace=True)

# Seleccionar los dos tipos con el mayor número de anuncios
top_vehicle_types = vehicle_type_stats.nlargest(2, 'num_ads')

# Graficar la dependencia de la cantidad de anuncios en cada tipo de vehículo
plt.figure(figsize=(10, 6))
sns.barplot(x='type', y='num_ads', data=top_vehicle_types)
plt.title('Número de Anuncios por Tipo de Vehículo')
plt.xlabel('Tipo de Vehículo')
plt.ylabel('Número de Anuncios')
plt.show()

# Mostrar la información sobre el número de anuncios y el precio promedio por tipo de v
print("Información sobre el número de anuncios y el precio promedio por tipo de vehícul
print(vehicle_type_stats)
```




```
Información sobre el número de anuncios y el precio promedio por tipo de vehículo:
           type         price  num_ads
0           SUV  11149.400000    12405
1           bus  17135.666667       24
2   convertible  14575.881166      446
3         coupe  14353.442901     2303
4     hatchback   6868.513849     1047
5      mini-van   8193.177433     1161
6       offroad  14292.294393      214
7         other  10989.714844      256
8        pickup  16057.410418     6988
9         sedan   6965.358647    12154
10        truck  16734.894924    12353
11          van  10546.941548      633
12        wagon   9088.134328     1541
```

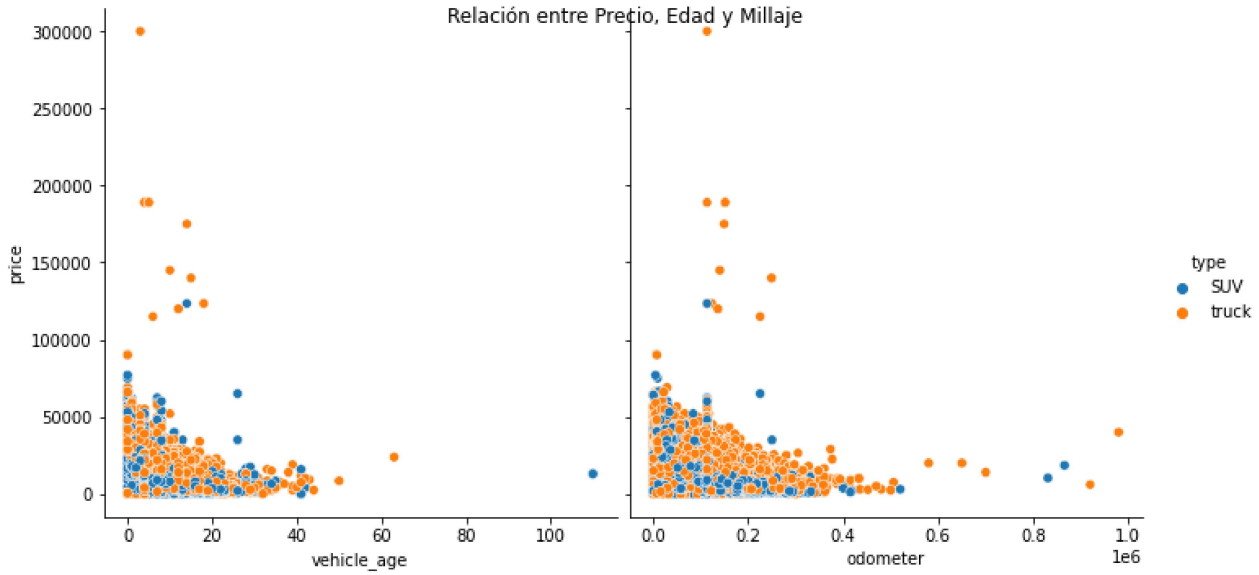
**Respuesta del estudiante** Los vehiculo con mayor precio por tipo fueron los $SUV$ y $truck$, como podemos observar el valor promedio y numero de ads de estos es la superior entre las demas.

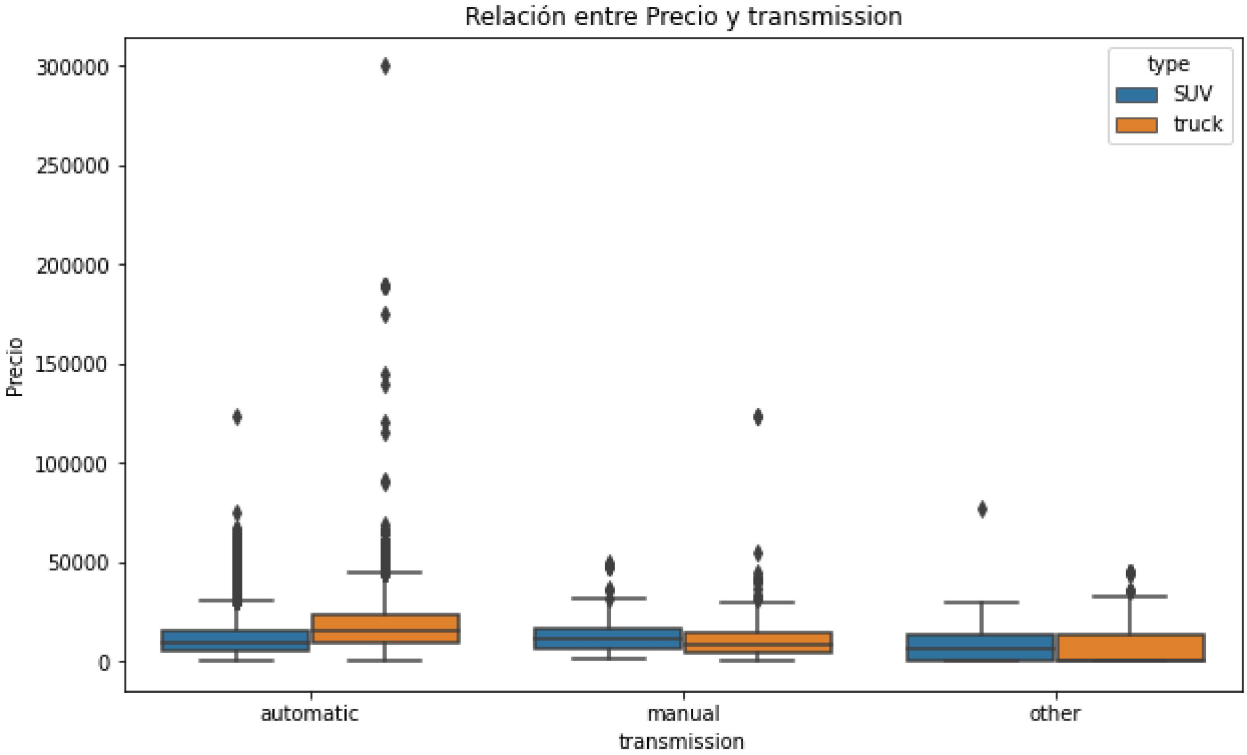
# Factores de precio

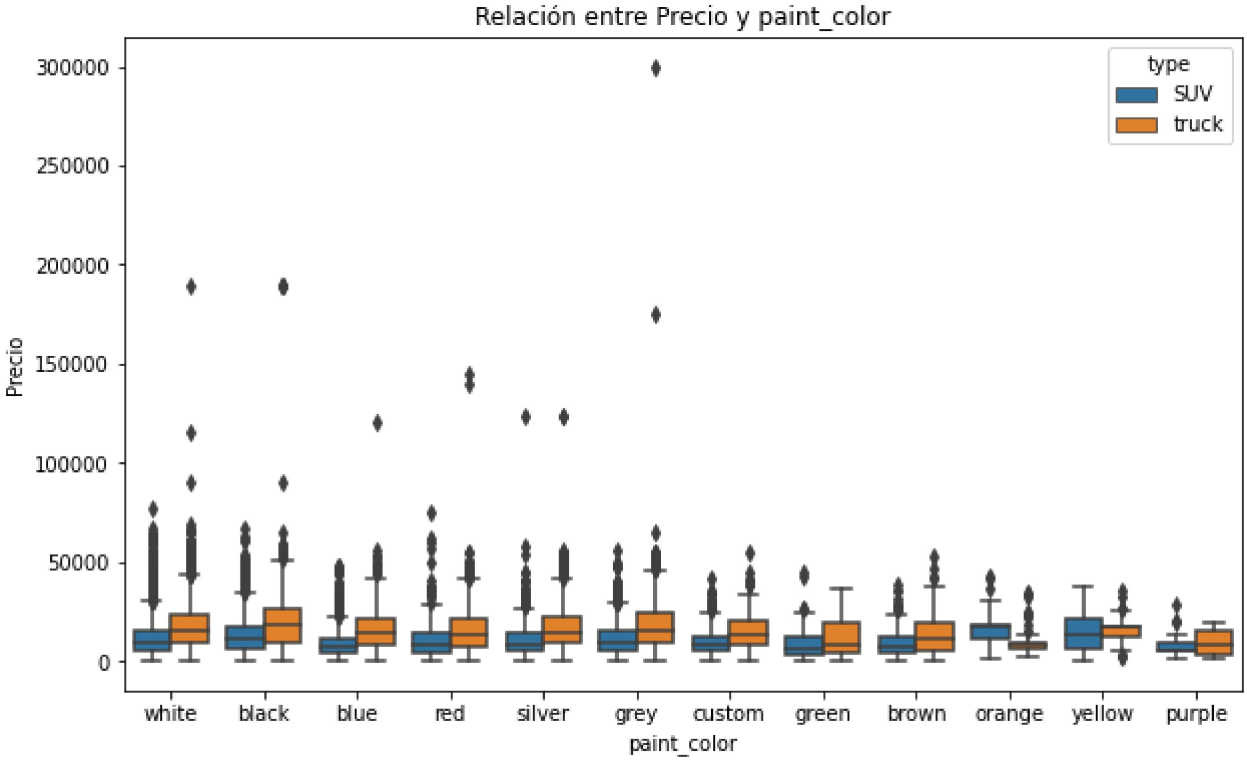
In [44]:

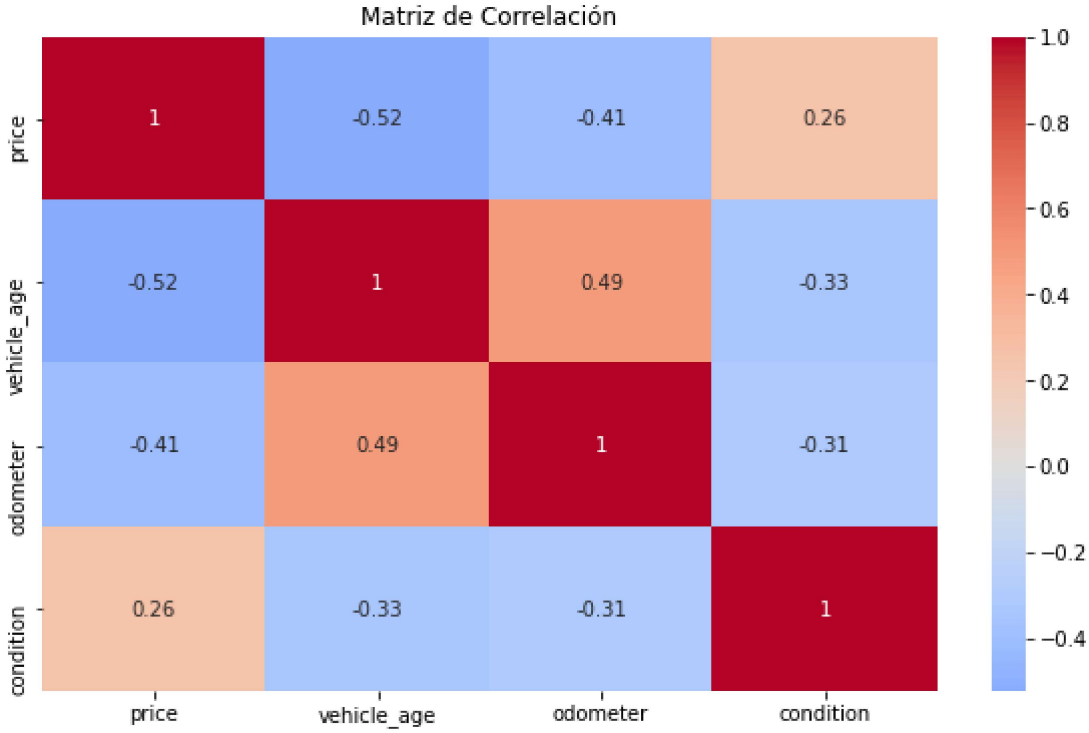
```
# Filtrar el DataFrame para los tipos de vehículos más populares

popular_vehicle_types = top_vehicle_types['type'].tolist()
filtered_df = df[df['type'].isin(popular_vehicle_types)]

# Crear un DataFrame con las variables seleccionadas
selected_columns = ['type', 'price', 'vehicle_age', 'odometer', 'condition', 'transmiss
selected_df = filtered_df[selected_columns]

# Filtrar las categorías con al menos 50 anuncios para las variables categóricas
categorical_vars = ['transmission', 'paint_color']
```

```
for var in categorical_vars:
    counts = selected_df[var].value_counts()
    selected_categories = counts[counts >= 50].index
    selected_df = selected_df[selected_df[var].isin(selected_categories)]

# Crear un gráfico de dispersión para las variables numéricas
numeric_vars = ['vehicle_age', 'odometer']
sns.pairplot(selected_df, x_vars=numeric_vars, y_vars='price', hue='type', height=5)
plt.suptitle('Relación entre Precio, Edad y Millaje')
plt.show()

# Crear gráficos de caja y bigotes para las variables categóricas
for var in categorical_vars:
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=selected_df, x=var, y='price', hue='type')
    plt.title(f'Relación entre Precio y {var}')
    plt.xlabel(var)
    plt.ylabel('Precio')
    plt.show()

# Calcular la matriz de correlación y mostrar un heatmap
correlation_matrix = selected_df.corr()
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Matriz de Correlación')
plt.show()
print(correlation_matrix)
```




Relación entre Precio y transmission
type
SUV
truck
Precio
300000
250000
200000
150000
100000
50000
0
automatic
manual
other
transmission


Relación entre Precio y paint_color
type
SUV
truck
Precio
300000
250000
200000
150000
100000
50000
0
white
black
blue
red
silver
grey
custom
green
brown
orange
yellow
purple
paint_color




```
               price  vehicle_age  odometer  condition
price        1.000000    -0.521983 -0.408729   0.259821
vehicle_age -0.521983     1.000000  0.486546  -0.333443
odometer    -0.408729     0.486546  1.000000  -0.305969
condition    0.259821    -0.333443 -0.305969   1.000000
```

**Respuesta del estudiante** En esta parte podemos ver la incidencia de las diferentes variables en el precio de los vehiculos, por tipo, año, millaje, condicion, color y transmision, podemos concluir que los vehiculos tipo $SUV$ o $TRUCK$ de color $blanco$ o $\neg ro$, de transmision $au \rightarrow matica$ de menos de $7$ años y que tengan menos de $50000$ millas son los vehiculos con mayores precios de nuestro dataframe, el valor de estos no depende de una sola variable en especifico, si no del conjunto en general es lo que se puede observar con una relacion no demasiado fuerte entre todas las variables ya mencionadas, pero a pesar de ello la relacion es evidente.

# Conclusión general

**Respuesta del estudiante** Como conclusiones podemos decir que la investigacion arrojo unos detalles interesantes, como el hecho de que un solo valor no determina el precio de un vehiculo, si no que esto se determina con un conjunto de valores los cuales son el $tipo$, , $transmision$, $edad$ y $millaje$, este conjunto nos arroja la informacion necesaria para saber que le da o quita valor a un vehiculo, como pudimos demostrar en los graficos contenidos en la investigacion, en el proceso gracias al procesado de los datos pude observar con detenimiento la fragil relacion que hay entre cada columna y el porque debia procesar los datos de la manera que lo hice, para asegurar que la investigacion fuera lo menos imparcial y conclusiva posible, como podemos ver se discrimino poca data en el proceso a la vez que se mejoro la que ya teniamos disponible, para poder lograr el cometido. Hablemos de los valores ausentes y atipicos con que nos encontramos en el dataframe que nos entregaron, estos valores que pudimos encontrar en el dataframe

simplemente parecen un grupo de errores que se acumularon en la captura de los datos, algo imposible de conocer por mi en el sentido de que no tengo acceso al equipo que se dedico a la captura para hacerles las preguntas correspondientes sobre estos errores, en el caso de los valores atipicos parecian mas un tipo de error de conversion de los datos, ya que habia bastantes datos en negativo en $price$ y $odometer$ principalmente, para solucionarlo, ya que no era una gran cantidad de data, como se puede observar, simplemente los deje fuera, la discriminacion fue bastante efectiva, ya que los valores no fueron influenciados de manera negativa y las conclusiones fueron mas que evidentes para poder llegar al entendimiento que tenemos ahora de la investigacion.