ACTIVIDAD FINAL - PYTHON PARA LA INTELIGENCIA ARTIFICIAL

Nombre: **Josseph Yaakob Catagua Cobos**
Enlace: GitHub

BASE DE DATOS: "ARTISTAS/PISTAS DESTACADAS DE SPOTIFY"

1. Introducción

Se usó la página Google Data Search para buscar de forma más rápida en las demás fuentes de información. Se obtuvo en *__keras__* la base de datos "Featured Spotify artists/tracks with metadata". Son muestras de datos de artistas y pistas dentro de la aplicación *Spotify* que aparecen en las listas de reproducción editoriales seleccionadas del 1 de Abril al 9 de Mayo del 2024.

Existen 3 archivos "*.csv*" anexados dentro de la carpeta SpotifyData, los cuales son:

- featured_Spotify_track_info.csv
  - Contiene las pistas destacadas, múltiples artistas, fechas y listas de reproducción separadas por comas. (Aproximadamente 15k de pistas únicas)
- featured_Spotify_artist_info.csv
  - Contiene artistas destacados, incluidas las repeticiones en listas de reproducción y fechas; siendo cada aparición asignada en una propia fila. Aparece un solo artista por cada pista, si existe una colaboración el artista fue escogido al azar. (Aproximadamente 10k de artistas únicos y 28k de filas)
- CLEANED_featured_Spotify_artist_info.csv
  - Similar al archivo "*featured_Spotify_artist_info.csv*" pero eliminando los valores Nulos y complementa los datos de género con datos extraidos de las biografías de Spotify.

2. Descripción de las columnas

- featured_Spotify_track_info.csv

Columna	Descripción	Tipo de dato
__ids__	Fecha en la que apareció el artista.	*str*
__names__	Nombre de la pista.	*str*
__popularity__	Métrica de popularidad definida por Spotify, cálculo no conocido.	*int*
__markets__	Código de mercado de los mercados en los que está disponible.	*int*
__artists__	ID de los artistas que crearon la pista.	*str* separados por ","
__release_date__	Fecha en la que se lanzó la pista.	*str* completa o solo año
__count__	Número de instancias separadas en las que apareció la pista.	*str*
__dates__	Fechas en las que la pista apareció en cualquier lista de reproducción.	*str*
__playlists_found__	Listas de reproducción editoriales en la que aparece la pista.	*str*
__duration_ms__	Duración de la pista.	*int* ms
__acousticness__	Medida de confianza de si la pista es acústica.	*float* $[0;1]$
__danceability__	Qué tan adecuado es una pista para bailar (tempo, estabilidad del ritmo y regularidad).	*float*
__energy__	Medida perceptiva de la intensidad y actividad (rápidas y ruidosas).	*float* $[0;1]$
__instrumentalness__	Probabilidad de que no contenga contenido vocal.	*float* $[0;1]$
__liveness__	Probabilidad de que se haya interpretado en vivo.	*float* $[0;1]$
__loudness__	Promedio del volumen general de la pista.	*float* $[-60;0]$ dB
__speechiness__	Probabilidad de que se detecte la presencia de palabras habladas en la pista.	*float* $[0;1]$
__tempo__	Promedio de la velocidad o ritmo de la pista.	*float* BPM
__valence__	Probabilidad de que suenen más positivas (felices, alegres, eufóricas, etc.).	*float* $[0;1]$

Columna	Descripción	Tipo de dato
__musicalkey__	Equivalente al campo "clave" en la sintaxis de API web de Spotify.	*int*
__musicalmode__	Equivalente al campo "modo" en la sintaxis de API web de Spotify.	*int* $[0;1]$
__time_signature__	Compás o tiempos estimado en cada compás.	*int* $[3;7]$

- featured_Spotify_artist_info.csv y CLEANED_featured_Spotify_artist_info.csv

Columna	Descripción	Tipo de dato
__dates__	Fecha en la que apareció el artista.	*str*
__ids__	ID único de Sotify de cada artista.	*str*
__nombres__	Nombre del artista de Spotify.	*str*
__monthly_listeners__	Media de oyentes mensuales de cada artista, recopilados entre Abril y Mayode 2024.	*float*
__popularity__	Métrica de popularidad definida por Spotify, cálculo no conocido.	*int*
__followers__	Número de seguidores que tiene el artista.	*int*
__genres__	Géneros musicales asociados a cada artista.	*str* separados por ","
__first_release__	Año del primer lamzamiento del artista.	*int*
__last_release__	Año del último lanzamiento del artista.	*int*
__num_releases__	Número total de lanzamientos que ha realizado el artista.	*int* $[0;20]$
__num_tracks__	Número de pistas del último álbum/sencillo que el artista a lanzado.	*int*
__playlists_found__	Lista de reproducción editorial en la que aparece el artista.	*str* separados por ","
__feat_track_ids__	ID de las pistas destacadas de Spotify.	*str*

3. Código Inicial

3.1. Imports

In [1]:
```
# imports necesarios para el archivo completo
import numpy as np
import pandas as pd
from os import path
import matplotlib.pyplot as plt
import seaborn as sns
```

3.2. Datos auxiliares

In [2]:
```
# Datos auxiliares para todo el archivo
plt.rcParams['font.family'] = 'Malgun Gothic' # Para evitar advertencias por caract

# Diccionario para cambiar a nombre del Mes {01: Enero, 02: Febrero ...}
list_month = ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agos
month_map = {str(i+1).zfill(2): month for i, month in enumerate(list_month)}
```

3.3. Obtener Data Frame

In [3]:
```
# Dirección de la base de datos .csv
clean_artists_url_csv = path.join('SpotifyData', 'CLEANED_featured_Spotify_artist_i
artists_url_csv = path.join('SpotifyData', 'featured_Spotify_artist_info.csv') # Da
tracks_url_csv = path.join('SpotifyData', 'featured_Spotify_track_info.csv') # Dato

# Cargar los datos de las tablas quitando NA
clean_artists_df = pd.read_csv(clean_artists_url_csv, sep=',', na_values='').dropna
artists_df = pd.read_csv(artists_url_csv, sep=',', na_values='').dropna()
tracks_df = pd.read_csv(tracks_url_csv, sep=',', na_values='').dropna()

# Mostrar información de los datos
print('\nINFO DF: Clean Artists\n')
clean_artists_df.info()
print('\nINFO DF: Artists\n')
artists_df.info()
print('\nINFO DF: Tracks\n')
tracks_df.info()
```

```
INFO DF: Clean Artists

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20251 entries, 0 to 20250
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   dates              20251 non-null  object 
 1   ids                20251 non-null  object 
 2   names              20251 non-null  object 
 3   monthly_listeners  20251 non-null  float64
 4   popularity         20251 non-null  int64  
 5   followers          20251 non-null  int64  
 6   genres             20251 non-null  object 
 7   first_release      20251 non-null  int64  
 8   last_release       20251 non-null  int64  
 9   num_releases       20251 non-null  int64  
 10  num_tracks         20251 non-null  int64  
 11  playlists_found    20251 non-null  object 
 12  feat_track_ids     20251 non-null  object 
dtypes: float64(1), int64(6), object(6)
memory usage: 2.0+ MB

INFO DF: Artists

<class 'pandas.core.frame.DataFrame'>
Index: 18647 entries, 0 to 27781
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   dates              18647 non-null  object 
 1   ids                18647 non-null  object 
 2   names              18647 non-null  object 
 3   monthly_listeners  18647 non-null  float64
 4   popularity         18647 non-null  int64  
 5   followers          18647 non-null  int64  
 6   genres             18647 non-null  object 
 7   first_release      18647 non-null  int64  
 8   last_release       18647 non-null  int64  
 9   num_releases       18647 non-null  int64  
 10  num_tracks         18647 non-null  int64  
 11  playlists_found    18647 non-null  object 
 12  feat_track_ids     18647 non-null  object 
dtypes: float64(1), int64(6), object(6)
memory usage: 2.0+ MB

INFO DF: Tracks

<class 'pandas.core.frame.DataFrame'>
Index: 15038 entries, 0 to 15051
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ids               15038 non-null  object 
 1   names             15038 non-null  object 
 2   popularity        15038 non-null  float64
```

```
 3   markets           15038 non-null  float64
 4   artists           15038 non-null  object
 5   release_date      15038 non-null  object
 6   duration_ms       15038 non-null  float64
 7   acousticness      15038 non-null  float64
 8   danceability      15038 non-null  float64
 9   energy            15038 non-null  float64
 10  instrumentalness  15038 non-null  float64
 11  liveness          15038 non-null  float64
 12  loudness          15038 non-null  float64
 13  speechiness       15038 non-null  float64
 14  tempo             15038 non-null  float64
 15  valence           15038 non-null  float64
 16  musicalkey        15038 non-null  float64
 17  musicalmode       15038 non-null  float64
 18  time_signature    15038 non-null  float64
 19  count             15038 non-null  float64
 20  dates             15038 non-null  object
 21  playlists_found   15038 non-null  object
dtypes: float64(16), object(6)
memory usage: 2.6+ MB
```

> Se puede observar que el número de datos en el archivo "*CLEANED_featured_Spotify_artist_info.csv*" es mayor que el archivo "*featured_Spotify_artist_info.csv*", además, el autor aseguró haber mejorado los datos. Por lo que nos aseguraremos de usar el primer documento para un análisis de datos mayor ya que ambos se basan en las mismas características o encabezados.

3.4. Arreglar y Eliminar Datos No Deseados

In [4]:
```
artists_df = clean_artists_df # Nos quedamos con el archivo con mayor cantidad de d

# Asegurar que no existan datos repetidos
artists_df.drop_duplicates(subset=artists_df.columns, keep='first')
tracks_df.drop_duplicates(subset=tracks_df.columns, keep='first')

# Mostrar información de los datos
print('\nINFO DF: Artists\n')
artists_df.info()
print('\nINFO DF: Tracks\n')
tracks_df.info()
```

```
INFO DF: Artists

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20251 entries, 0 to 20250
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   dates              20251 non-null  object 
 1   ids                20251 non-null  object 
 2   names              20251 non-null  object 
 3   monthly_listeners  20251 non-null  float64
 4   popularity         20251 non-null  int64  
 5   followers          20251 non-null  int64  
 6   genres             20251 non-null  object 
 7   first_release      20251 non-null  int64  
 8   last_release       20251 non-null  int64  
 9   num_releases       20251 non-null  int64  
 10  num_tracks         20251 non-null  int64  
 11  playlists_found    20251 non-null  object 
 12  feat_track_ids     20251 non-null  object 
dtypes: float64(1), int64(6), object(6)
memory usage: 2.0+ MB

INFO DF: Tracks

<class 'pandas.core.frame.DataFrame'>
Index: 15038 entries, 0 to 15051
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ids               15038 non-null  object 
 1   names             15038 non-null  object 
 2   popularity        15038 non-null  float64
 3   markets           15038 non-null  float64
 4   artists           15038 non-null  object 
 5   release_date      15038 non-null  object 
 6   duration_ms       15038 non-null  float64
 7   acousticness      15038 non-null  float64
 8   danceability      15038 non-null  float64
 9   energy            15038 non-null  float64
 10  instrumentalness  15038 non-null  float64
 11  liveness          15038 non-null  float64
 12  loudness          15038 non-null  float64
 13  speechiness       15038 non-null  float64
 14  tempo             15038 non-null  float64
 15  valence           15038 non-null  float64
 16  musicalkey        15038 non-null  float64
 17  musicalmode       15038 non-null  float64
 18  time_signature    15038 non-null  float64
 19  count             15038 non-null  float64
 20  dates             15038 non-null  object 
 21  playlists_found   15038 non-null  object 
dtypes: float64(16), object(6)
memory usage: 2.6+ MB
```

4. Análisis De Datos

In [5]:
```
top_n = 3 # Editable, numero del top a conciderar menor a 10 (Conclusiones basadas

# Generación de colores para graficar
palette = sns.color_palette("husl", top_n)
palette.append('#808080')
display(palette)
```

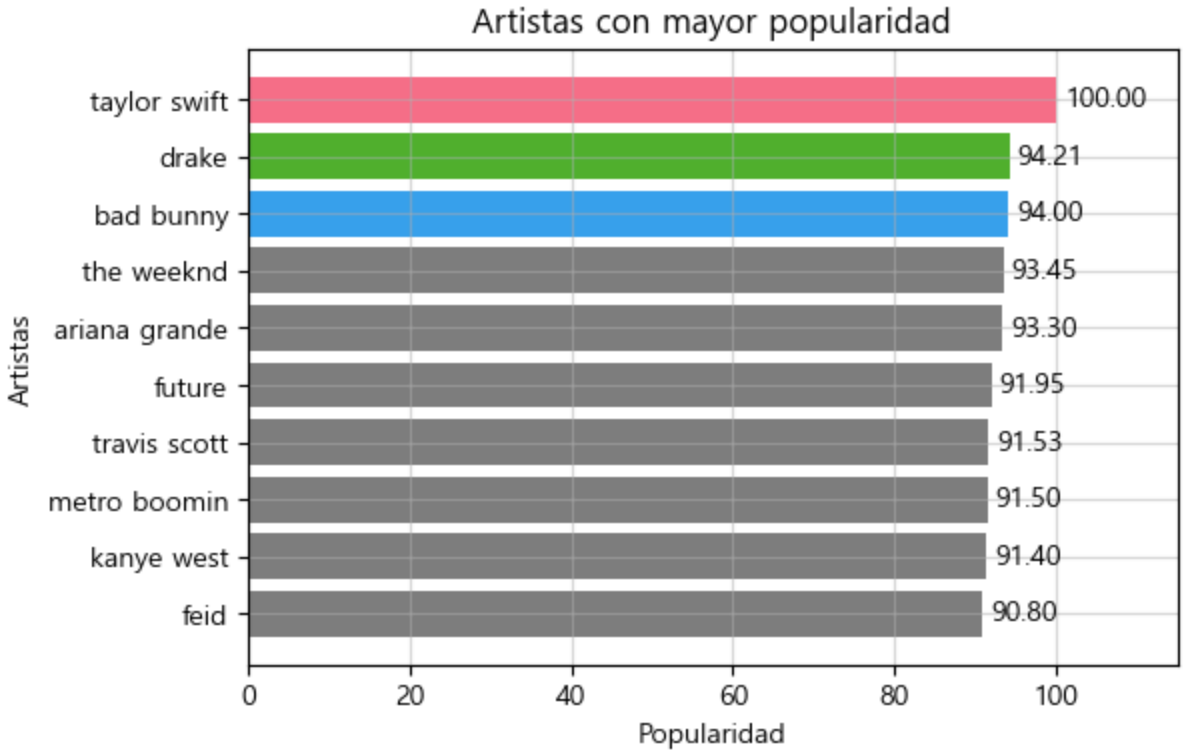
> Declaramos el número de artistas a analizar, por defecto las conclusiones de cada apartado o gráfica serán bajo el top 3 de artistas y/o pistas.

4.1. Análisis De Popularidad

4.1.1. Artistas con mayor popularidad

¿Cuál es el `top_n` de los artistas según su popularidad?

In [6]:
```
# Escogemos los datos a analizar, para no modificar la tabla
popularity_df = artists_df[['names', 'popularity']].copy() # Nombres y Popularidad

# Agrupamos ya que hay valores repetidos y obtenemos los 10 valores más altos
top_artists = popularity_df.groupby('names')['popularity'].mean().nlargest(10).rese

fig, ax = plt.subplots(figsize=(6, 4))

# Lista de colores para diferencia el top a analizar
colors = palette[:-1] + [palette[-1]]*(len(top_artists['names'])-top_n)

# Graficamos por popularidad en X y los nombres de los artistas en Y
ax.barh(top_artists['names'], top_artists['popularity'], color=colors)

# Mostrar el valor de popularidad en la barra
for i, v in enumerate(top_artists['popularity']):
    ax.text(v + 1, i, f'{v:.2f}', va='center')

ax.set_title('Artistas con mayor popularidad')
ax.set_xlabel('Popularidad')
ax.set_ylabel('Artistas')
ax.set_xlim(0, 115) # La popularidad tiene un valor maximo de 100
ax.invert_yaxis() # Para observar el máximo arriba
ax.grid(alpha=0.5)
plt.show()
```



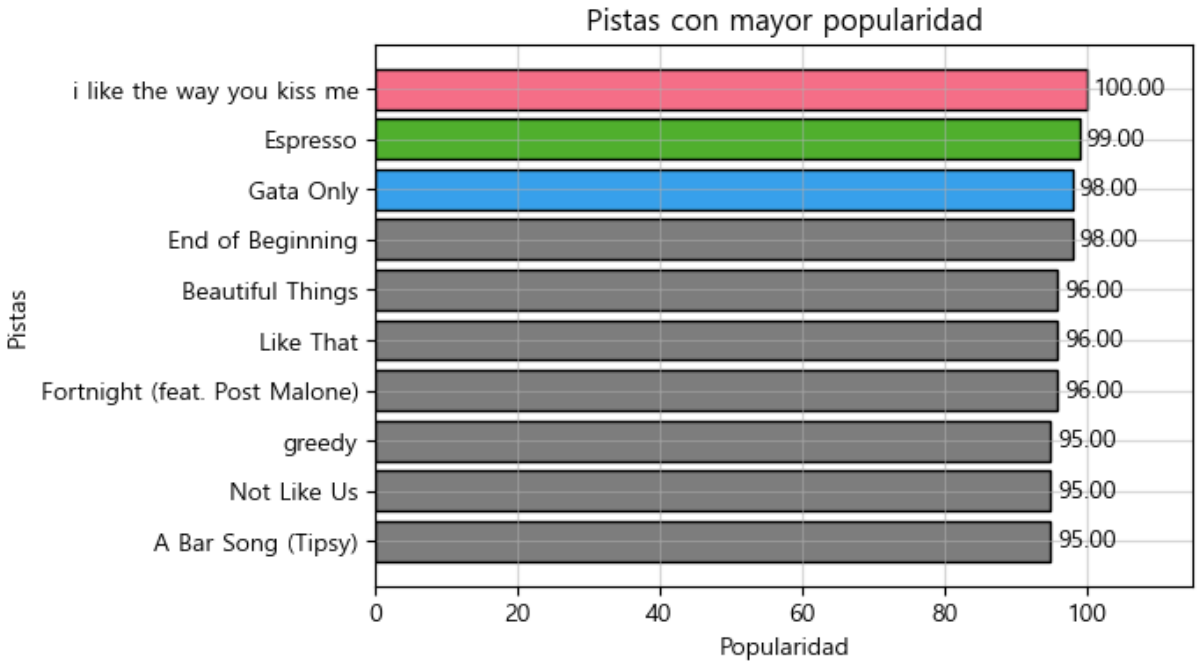
> El artista con mayor popularidad según la base de datos basada en Spotify es "*__Taylor Swift__" *con el máximo porcentaje de popularidad (100%), seguido de* "__Drake__" *con un valor del 94.21% y por último* "__Bad Bunny__*" con una popularidad del 94%.

4.1.2. Pistas con mayor popularidad

¿Cuál es el `top_n` de las pistas según su popularidad?

In [7]:
```
# Escogemos los datos a analizar, para no modificar la tabla
popularity_df = tracks_df[['names', 'popularity']].copy() # Nombres y Popularidad

# No hay valores repetidos y obtenemos los 10 valores más altos
top_tracks = popularity_df.sort_values('popularity', ascending=False).head(10).rese

fig, ax = plt.subplots(figsize=(6, 4))

# Graficamos por popularidad en X y los nombres de las pistas en Y
ax.barh(top_tracks['names'], top_tracks['popularity'], edgecolor='black', color=col

# Mostrar el valor de popularidad en la barra
for i, v in enumerate(top_tracks['popularity']):
    ax.text(v + 1, i, f'{v:.2f}', va='center')

ax.set_title('Pistas con mayor popularidad')
ax.set_xlabel('Popularidad')
ax.set_ylabel('Pistas')
ax.set_xlim(0, 115) # La popularidad tiene un valor maximo de 100
ax.invert_yaxis() # Para observar el máximo arriba
```

```
ax.grid(alpha=0.5)
plt.show()
```



> La pista con mayor popularidad según la base de datos basada en Spotify es "*__I like the way you kiss me__" *con el máximo porcentaje de popularidad (100%), seguido de* "__Espresso__" *con un valor de 99% y por último* "__Gata Only__*" con una popularidad del 98%.

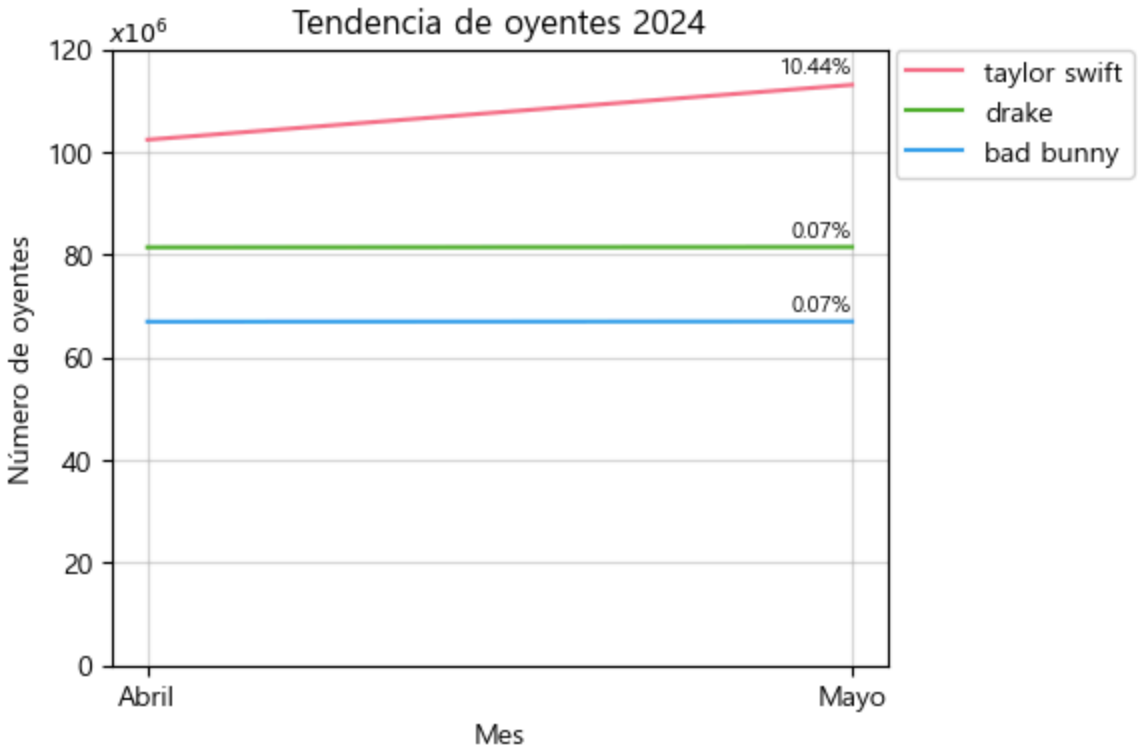
4.2. Tendencias En Oyentes Mensuales

4.2.1. Crecimiento de oyentes

¿Cuál es el crecimiento, por cada mes en 2024, de oyentes del `top_n` de artistas?

In [8]:
```
# Obtenemos los nombres del top
n_artists = top_artists.head(top_n)['names'].unique()

# Tomamos los datos a analizar, para no modificar la tabla
listeners_df = artists_df[['names', 'dates', 'monthly_listeners']].copy() # Nombres
listeners_df = listeners_df[listeners_df['names'].isin(n_artists)].reset_index(drop
listeners_df[['years', 'months', 'days']] = listeners_df['dates'].str.split('-', ex
listeners_df = listeners_df.drop(['dates', 'years'], axis=1) # Elimino dates y year

# Conversiones de dato
listeners_df['monthly_listeners'] = listeners_df['monthly_listeners']/1000000 # Val
listeners_df['months'] = listeners_df['months'].map(month_map) # Cambiamos los índi

fig, ax = plt.subplots(figsize=(5, 4))

# Agrupamos por nombre y mes, media de valores de visualizaciones
df = listeners_df.groupby(['names', 'months'])['monthly_listeners'].mean().reset_in
```

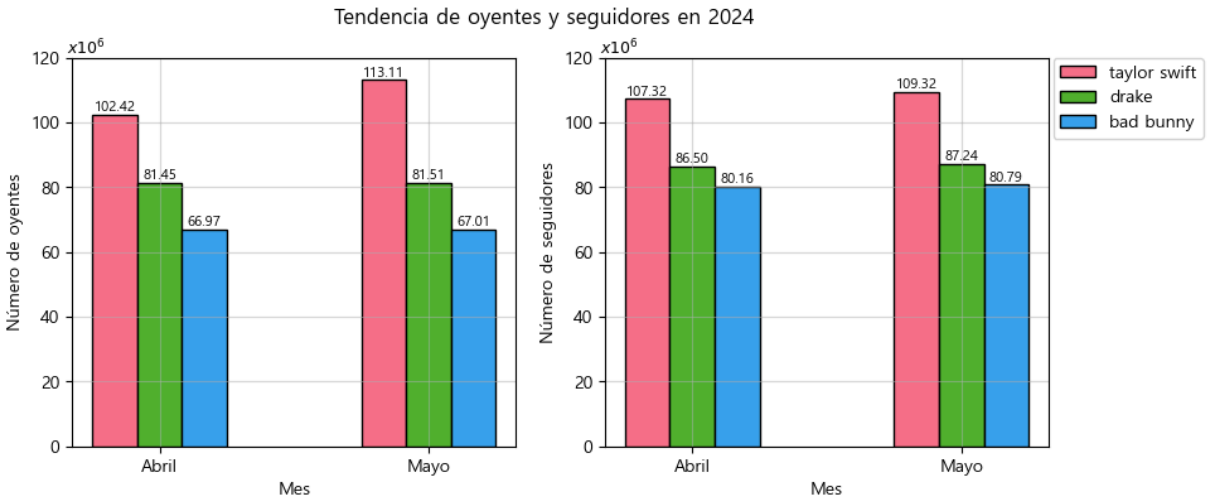
```
# Por cada mes se recorren los n artistas
for i, j in enumerate(n_artists):
    j_df = df[df['names'] == j][['months', 'monthly_listeners']] # Por cada artista
    ax.plot(j_df['months'], j_df['monthly_listeners'], label=j, color=palette[i]) #
    percent = ((j_df['monthly_listeners'].values[-1] - j_df['monthly_listeners'].va
    ax.text(j_df['months'].values[-1], j_df['monthly_listeners'].values[-1] + 1, f'

# Ubicar la legenda fuera de la grafica
ax.legend(bbox_to_anchor=(1.01, 1),
            loc='upper left', # Referencia la esquina superior izquierda de la lege
            borderaxespad=0.) # Espacios fuera del borde

# Colocal en el eje, la magnitud de la escala
ax.annotate(r'$x10^6$', xy=(0.075, 1), xycoords='axes fraction', xytext=(0, 0),
            textcoords='offset points', ha='right', va='bottom', fontsize=9)

plt.title(f'Tendencia de oyentes 2024')
ax.set_ylabel('Número de oyentes')
ax.set_xlabel('Mes')
ax.set_ylim(0, 120) # Los oyentes tiene un valor maximo de 110
ax.grid(alpha=0.5)
plt.show()
```



> Para los tres artistas en el top, poseen un crecimiento mensual en el número de oyentes según Spotify. "***Taylor Swift***" *tiene un crecimiento del 10.44% aproximado entre Abril y Mayo,* "**Drake**" *y* "**Bad Bunny***" crecen en oyentes a un aproximado de 0.07% mensual.

2.2.1. Comparación de seguidores y oyentes mensuales: Ver la relación entre el número de seguidores y los oyentes mensuales.

In [9]:
```
# Tomamos los datos a analizar, para no modificar la tabla
followers_df = artists_df[['names', 'dates', 'followers', 'monthly_listeners']].cop
followers_df = followers_df[followers_df['names'].isin(n_artists)].reset_index(drop
followers_df[['years', 'months', 'days']] = followers_df['dates'].str.split('-', ex
followers_df = followers_df.drop(['dates', 'years'], axis=1) # Elimino dates y year

# Conversiones de dato
followers_df['monthly_listeners'] = followers_df['monthly_listeners']/1000000 # Val
followers_df['followers'] = followers_df['followers']/1000000 # Valores muy grandes
followers_df['months'] = followers_df['months'].map(month_map) # Cambiamos los índi

fig, ax = plt.subplots(1,2,figsize=(10, 4), sharex=True)

# Agrupamos por nombre y mes, media de valores de visualizaciones
df = followers_df.groupby(['names', 'months'])[['monthly_listeners', 'followers']].
# Reorganizar. Meses como indices, Names como subcolumnas para Oyentes y Seguidores
data = df.pivot_table(index='months', columns='names', values=['monthly_listeners',
months = data.index # Meses
width = 0.5 / len(n_artists) # Calcular el ancho de las barras

# Grafica las barras correspondientes. Artista por mes en X y Numero de Oyentes y S
for i, artist in enumerate(n_artists):
    x = np.arange(len(months)) + i * width
    y = data[('monthly_listeners', artist)]
    ax[0].bar(x, y, width=width, align='center', color=palette[i], edgecolor='black
    for xi, yi in zip(x, y):
        ax[0].text(xi, yi, f'{yi:.2f}', va='bottom', ha='center', fontsize=8)
    y = data[('followers', artist)]
    ax[1].bar(x, y, width=width, align='center', color=palette[i], edgecolor='black
    for xi, yi in zip(x, y):
        ax[1].text(xi, yi, f'{yi:.2f}', va='bottom', ha='center', fontsize=8)

fig.suptitle('Tendencia de oyentes y seguidores en 2024')
ax[0].set_ylabel('Número de oyentes')
ax[1].set_ylabel('Número de seguidores')

# Modificación por gráfica
for i in range(len(ax)):
    ax[i].set_xlabel('Mes')
    ax[i].set_xticks([r + width for r in range(len(months))])
    ax[i].set_xticklabels(months)
    ax[i].grid(alpha=0.5)
    ax[i].set_ylim(0, 120)
    # Colocal en el eje, la magnitud de la escala
    ax[i].annotate(r'$x10^6$', xy=(0.075, 1), xycoords='axes fraction', xytext=(0,
                   textcoords='offset points', ha='right', va='bottom', fontsize=9)

ax[1].legend(bbox_to_anchor=(1.01, 1), # Ubicar la legenda fuera de la grafica
             loc='upper left', # Referencia la esquina superior izquierda de la lege
             borderaxespad=0.) # Espacios fuera del borde
plt.show()
```



> Para los tres artistas en el top, poseen un crecimiento mensual en el número de oyentes según Spotify. "*__Taylor Swift__" *tiene un crecimiento de aproximadamente $10.89x10^6$ en oyentes y $2x10^6$ en seguidores.* "__Drake__" *posee un aumento aproximado en oyentes de $0.06x10^6$ y en seguidores de $0.74x10^6$. Por ultimo,* "__Bad Bunny__*" posee un crecimiento aproximado en oyentes de $1.04x10^6$ y en seguidores de $0.63x10^6$.

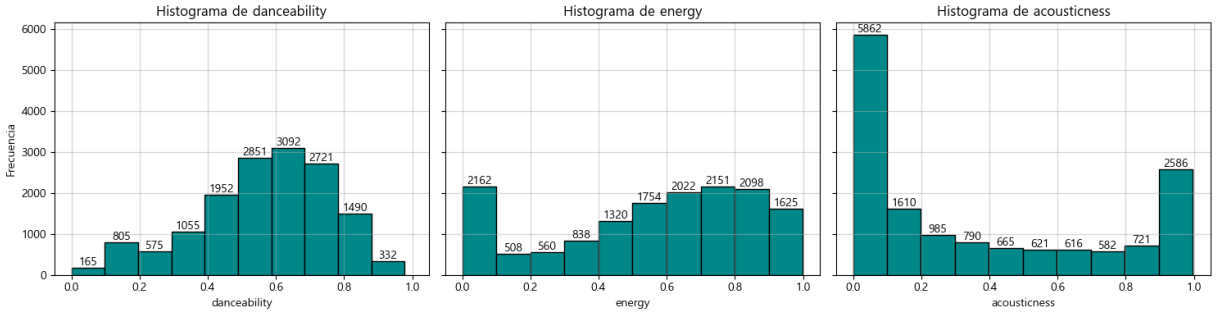
2.3. Características de Canciones

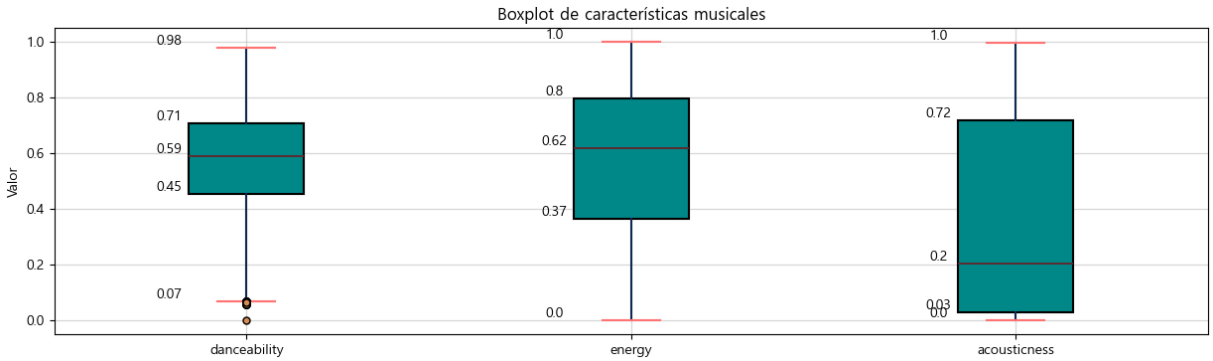
2.3.1. Distribución de características musicales

¿Cuál es la frecuencia en las características musicales para cada pista?

In [10]:
```
# Tomamos los datos a analizar, para no modificar la tabla
features = ['danceability', 'energy', 'acousticness']

# Subplots dependientes del número de características a analizar
fig, ax = plt.subplots(1, len(features), figsize=(5*len(features), 4), sharex=True,

# Graficar
for i, feature in enumerate(features, 1):
    _, _, patches = ax[i-1].hist(tracks_df[feature], bins=10, color='darkcyan', edg
    ax[i-1].set_title(f'Histograma de {feature}')
    ax[i-1].set_xlabel(feature)
    ax[i-1].grid(alpha=0.5)
    # Colocar valores sobre la barra
    for j in range(len(patches)):
        ax[i-1].text(patches[j].get_x() + patches[j].get_width() / 2, patches[j].ge
                     int(patches[j].get_height()), ha='center', va='bottom', fontsi

ax[0].set_ylabel('Frecuencia')
plt.tight_layout() # Evita que las etiquetas, títulos y bordes de cada subplot se s
plt.show()
```

```
# Crear un boxplot para cada característica

plt.figure(figsize=(5*len(features), 4))

# Modificador de colores
boxprops = dict(color="black", facecolor="darkcyan", linewidth=1.5)
medianprops = dict(color="#5c2f33", linewidth=1.5)
whiskerprops = dict(color="#102c54", linewidth=1.5)
capprops = dict(color="#ff7474", linewidth=1.5)
flierprops = dict(markerfacecolor="#d99058", marker="o", markersize=5, linestyle='n

# Graficar los datos en un bloxpot
bp = tracks_df[features].boxplot(boxprops=boxprops, medianprops=medianprops, whiske

# Obtener y escribir los valores en los puntos correspondientes
stats = tracks_df[features].describe().transpose()
for i, feature in enumerate(features, 1):
    q1, q2, q3 = stats.loc[feature, ['25%', '50%', '75%']] # Obtener los valores de
    IQR = q3 - q1
    min_f = stats.loc[feature, 'min']
    max_f = stats.loc[feature, 'max']
    min_f = min_f if q1 - 1.5 * IQR < min_f else q1 - 1.5 * IQR
    max_f = max_f if q3 + 1.5 * IQR > max_f else q3 + 1.5 * IQR
    offset = 0.2
    plt.text(i-offset, q1, str(round(q1, 2)), ha='center', va='bottom', color='blac
    plt.text(i-offset, q2, str(round(q2, 2)), ha='center', va='bottom', color='blac
    plt.text(i-offset, q3, str(round(q3, 2)), ha='center', va='bottom', color='blac
    plt.text(i-offset, min_f, str(round(min_f, 2)), ha='center', va='bottom', color
    plt.text(i-offset, max_f, str(round(max_f, 2)), ha='center', va='bottom', color

plt.title('Boxplot de características musicales')
plt.ylabel('Valor')
plt.grid(alpha=0.5)
plt.xticks(range(1, len(features) + 1), features)
plt.show()
```





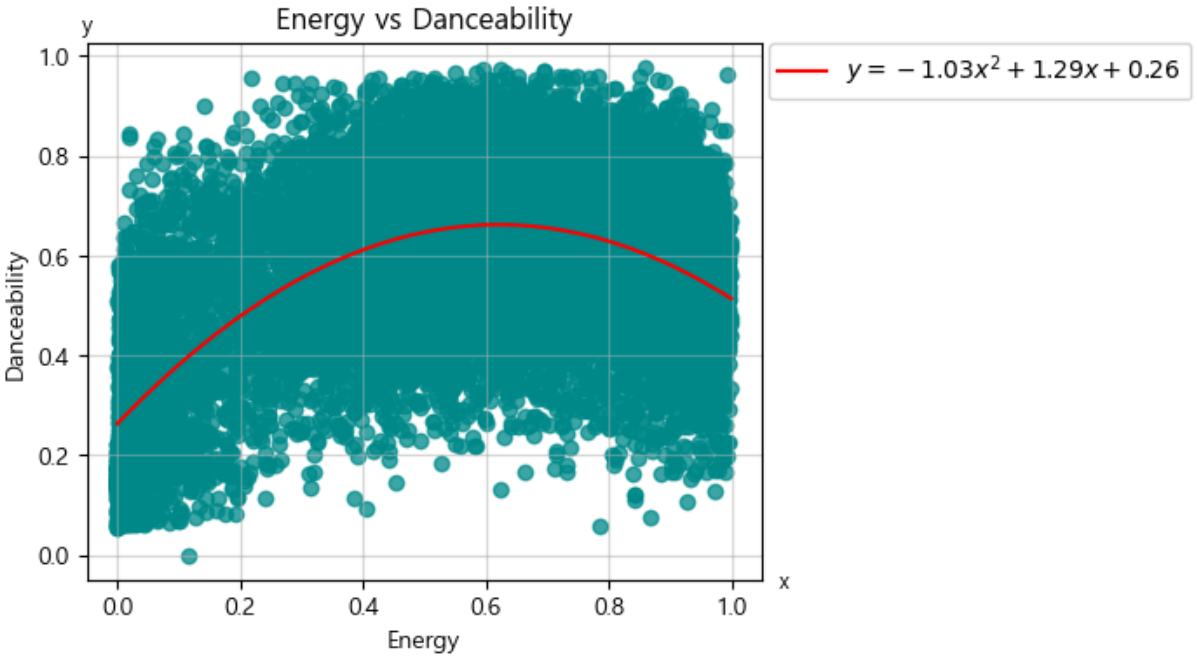
> En los diagramas se puede observar que las pistas analizadas son más probables que sean término medio en ser bailables o no, que no sean enérgicos o que casi lo sean y que no sean acústicas.

2.3.2. Relación entre características

¿Cuál es la tendencia entre las características de que tan enérgica y bailable es una pista?

In [11]:
```
from scipy.optimize import curve_fit

# Datos
x = tracks_df['energy'].copy()
y = tracks_df['danceability'].copy()

# Scatter plot
plt.figure(figsize=(5, 4))
plt.scatter(x, y, alpha=0.75, color='darkcyan')
plt.title('Energy vs Danceability')
plt.xlabel('Energy')
plt.ylabel('Danceability')
plt.grid(alpha=0.5)

# Definimos una función
def func(x, a, b, c):
    return [a * x0**2 + b*x0 + c for x0 in x]

# Ajuste de curva
popt, _ = curve_fit(func, x, y)
a, b, c = popt

# Graficar función
x_fit = np.linspace(min(x), max(x), 100)
plt.plot(x_fit, func(x_fit, *popt), color='red',
         label=r'$y = {:.2f}x^2 + {:.2f}x + {:.2f}$'.format(a, b, c),
         linewidth=1.5)  # Especifica el ancho de la línea

# Indicadores de cual es X y cual es Y en la ecuación
plt.annotate(r'y', xy=(0.01, 1.01), xycoords='axes fraction', xytext=(0, 0),
                   textcoords='offset points', ha='right', va='bottom', fontsize=9)
plt.annotate(r'x', xy=(1.04, -0.025), xycoords='axes fraction', xytext=(0, 0),
                   textcoords='offset points', ha='right', va='bottom', fontsize=9)
```

```
# Leyenda fuera del gráfico
plt.legend(bbox_to_anchor=(1.01, 1), # Ubicar la legenda fuera de la grafica
           loc='upper left', # Referencia la esquina superior izquierda de la lege
           borderaxespad=0.) # Espacios fuera del borde

plt.show()
```



> Según los datos de Spotify, entre que tan enérgico y que tan bailable es, podemos observar que para que una pista sea muy bailable tiene que tener un aproximado de nivel medio en la energía que transmite (0.6). Por ejemplo el Metal, aunque expone mucha energía puede no ser bailable para algunas personas. También, podemos analizar que tiene una pendiente positiva (crecimiento) antes de 0.6 loq ue indica un aproporcionalidad directa entre ambas características.

2.4. Análisis Temporal

2.4.1. Tendencias de lanzamientos

¿Cuál es la tendencia de lanzamientos por cada año?

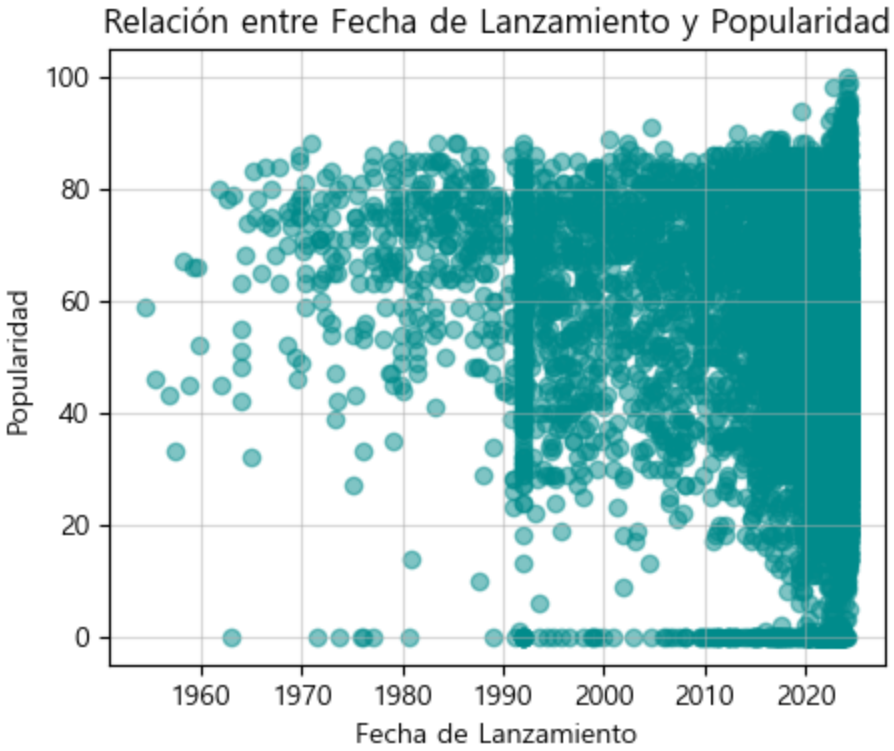
In [12]:
```
# Tomamos los datos a analizar, para no modificar la tabla
release_df = tracks_df[['release_date', 'popularity']].copy()
release_df['release_date'] = pd.to_datetime(release_df['release_date'], errors='coe
release_df['release_date'] = release_df['release_date'].fillna(pd.to_datetime('1992

# Contamos cuantos lanzamientos hay por año
release_counts = release_df['release_date'].dt.year.value_counts().sort_index()

#Graficamos los puntos
```

```
plt.figure(figsize=(5, 4))
plt.plot(release_counts.index, release_counts.values, marker='o', color='darkcyan')
plt.title('Número de Lanzamientos por Año')
plt.xlabel('Año')
plt.ylabel('Número de Lanzamientos')
plt.grid(alpha=0.5)
plt.show()
```

> Es normal tener una tendencia de este tipo ya que la aplicación fue creada en el 2006 y lanzada el 2008 dando una oportunidad de tener una biblioteca de música como servicio y también el promocionar la música por internet. Se puede observar que desde aproximadamente 2011 se popularizó y hubo un crecimiento exponencial.

¿Cuál es la relación entre el lanzamiento por cada año y la popularidad de las pistas?

In [13]:
```
# Graficamos los puntos entre la popularidad por cada lanzamiento
plt.figure(figsize=(5, 4))
plt.scatter(release_df['release_date'], release_df['popularity'], alpha=0.5, color=

# Graficamos

plt.title('Relación entre Fecha de Lanzamiento y Popularidad')
plt.xlabel('Fecha de Lanzamiento')
plt.ylabel('Popularidad')
plt.grid(alpha=0.5)
plt.show()
```



> Podemos ver un crecimiento significativo en la popularidad de algunas pistas lanzadas después del 2020 dando a entender un mayor consumo de este servicio como de la calidad de pistas que se producen. De la misma manera, se puede observar un crecimiento de personas que publican o exponen sus pistas, debido a la gran cantidad de puntos o pistas en ese área.

Fin