

Avance en la utilización de la API de spotify como base del proyecto.

* En este transcurso tuvimos algunos problemas para conseguir el Token pero con el paso de las clases, lo automatizamos.

(Imágenes de código y mejoras)

```
1 import requests
2
3 ▼ def get_access_token(client_id,
4   client_secret):
5     token_url =
6       "https://accounts.spotify.com/api/token"
7
8 ▼   headers = {
9     "Content-Type": "application/x-www-
10  form-urlencoded"
11   }
12
13   data = {
14     "grant_type": "client_credentials",
15     "client_id": client_id,
16     "client_secret": client_secret
17   }
18
19   response = requests.post(token_url,
20     headers=headers, data=data)
21
22   if response.status_code == 200:
23     return response.json()
24   else:
25     raise Exception(f"Error al obtener
26 el token: {response.status_code} -
27 {response.text}")
28
```

```
1 import requests
2
3 ▼ def get_access_token(client_id,
4   client_secret):
5     token_url =
6       "https://accounts.spotify.com/api/token"
7
8 ▼   headers = {
9     "Content-Type": "application/x-www-
10  form-urlencoded"
11   }
12
13   data = {
14     "grant_type": "client_credentials",
15     "client_id": client_id,
16     "client_secret": client_secret
17   }
18
19   response = requests.post(token_url,
20     headers=headers, data=data)
```

```
21
22
23
24   client_id =
25     '8fe819ee033e4a6a86ebac251c4245c4'
26   client_secret =
27     '1f53f4f2543b459a90869e3a1cae633a'
28
29 ▼ try:
30   token = get_access_token(client_id,
31     client_secret)
32   print("Access Token:", token)
33
34 ▼ except Exception as e:
35   print(e)
```

Empezamos con las funciones a utilizar próximamente en el bot. Artista y relacionados.

*No tuvimos mayores problemas con estas implementaciones. Únicamente que además de esto había que agregarle valor.

(Imágenes de código y mejoras)

```
70 ▼     if response.status_code == 200:
71         return response.json()
72 ▼     else:
73         raise Exception(f"Error al obtener
74             artistas relacionados:
75             {response.status_code} - {response.text}")
76 # Ejemplo de uso
77 client_id =
78     '8fe819ee033e4a6a86ebac251c4245c4'
79 client_secret =
80     '1f53f4f2543b459a90869e3a1cae633a1'
81
82 ▼ try:
83     token = get_access_token(client_id,
84     client_secret)
85     artist_name = input("Ingresa el nombre
86         de un artista: ")
87     related_artists =
88     get_related_artists(token, artist_name)
89
90     print("Artistas relacionados:")
91     for artist in
92         related_artists['artists']:
93         print(f"- {artist['name']}")
94 ▼ except Exception as e:
95     print(e)
```

```
55
56 ▼ try:
57     token = get_access_token(client_id,
58     client_secret)
59     artist_name = input("Ingresa el nombre
60         de un artista: ")
61     related_artists =
62     get_related_artists(token, artist_name)
63
64     print("Artistas relacionados:")
65     for artist in
66         related_artists['artists']:
67         print(f"- {artist['name']}")
68 ▼ except Exception as e:
69     print(e)
```

```
32     access_token =
33     'BQCaXvF68EIqLCG0wgC5ns0De_ChtfM-
34     9d5o3aFQ3sHlNLq0T0278UMdw5K6X86C8nW8fU0egjI-
35     pfYwNdUoE89YtzcqjX0mZDr4rxKzoTRi7-61zzc'
36     artist_id = '19HM5j0ULGSmEoRcrSe5x3' # ID
37     del artista
38
39
40     print("Artistas relacionados:")
41 ▼     for artist in
42         related_artists['artists']:
43             print(f"- {artist['name']}")
44     except Exception as e:
45         print(e)
46
47     def get_artist_id(access_token,
48     artist_name):
49
50         search_url =
51         f"https://api.spotify.com/v1/search?q=
52             {artist_name}&type=artist"
53
54     headers = {
55         "Authorization": f"Bearer
56             {access_token}"
57
58         response = requests.get(search_url,
59         headers=headers)
60
61         if response.status_code == 200:
62             artists = response.json()['artists']
63             ['items']
64
65             if artists:
66                 return artists[0]['id'] #
67                 Devuelve el ID del primer artista encontrado
68             else:
69                 raise Exception("Artista no
70                 encontrado.")
71             else:
72                 raise Exception(f"Error al buscar
73                     el artista: {response.status_code} -
74                     {response.text}")
75
76     def get_related_artists(access_token,
77     artist_name):
78
79         artist_id = get_artist_id(access_token,
80         artist_name)
81         url =
82         f"https://api.spotify.com/v1/artists/{artist
83         _id}/related-artists"
84
85         headers = {
86             "Authorization": f"Bearer
87             {access_token}"
88
89         response = requests.get(url,
90         headers=headers)
```

Como nos faltaba contenido de valor, optamos por diferentes alternativas. Una de ellas fue el armado de una trivia interactiva para que los usuarios se quedan. (Al poco tiempo fue rechazada)

*Aregar contenido de valor fue una problemática que nos acompañó hasta el final del proyecto

(Imágenes de código y mejoras)

```
97 # Ejemplo de uso
98 client_id =
'8fe819ee033e4a6a86ebac251c4245c4'
99 client_
100
101▼ try:
102     token = get_access_token(client_id,
client_secret)
103     artist_name = input("Ingresa el nombre
de un artista: ")
104     related_artists =
get_related_artists(token, artist_name)
105
106     print("Artistas relacionados:")
107▼   for artist in
related_artists['artists']:
108         print(f"- {artist['name']}")#
-----#
111 # Arranque del juego de trivia
112 trivia_game(token, related_artists)
113 # -----#
115▼ except Exception as e:
116     print(e)
117
```

INCLUSO intentamos utilizar metacritic para poder acceder a su base de dato y que desde aquí nos envíen reseñas. (NO LO LOGRAMOS Y DESCARTAMOS)

```
76▼ def trivia_game(access_token,
related_artists):
77     correct_artist =
random.choice(related_artists['artists'])
78     correct_answer = correct_artist['name']
79
80     print(f"¿Quién de estos es un artista
relacionado con {correct_answer}?")
81
82     options = [correct_answer] +
random.sample([artist['name'] for artist
in related_artists['artists']] if
artist['name'] != correct_answer, 3)
83     random.shuffle(options)
84
85▼   for i, option in enumerate(options):
86     print(f"{i + 1}. {option}")
87
88▼   try:
89     answer = int(input("Selecciona el
número de tu respuesta: ")) - 1
90▼     if options[answer] ==
correct_answer:
91         print("¡Correcto!")
92▼     else:
93         print(f"Incorrecto. La
respuesta correcta era {correct_answer}.")
94▼   except (ValueError, IndexError):
95     print("Por favor, selecciona un
número válido.")
```

```
86▼ def get_music_reviews(options):
87▼   try:
88     # Aquí deberías colocar la URL de
la API que proporciona las reseñas
89     url =
"https://static.metacritic.com/"
90     response = requests.get(url,
params=options)
91     response.raise_for_status()
92     return response.json()
93▼   except
requests.exceptions.RequestException as e:
94     return str(e)
```

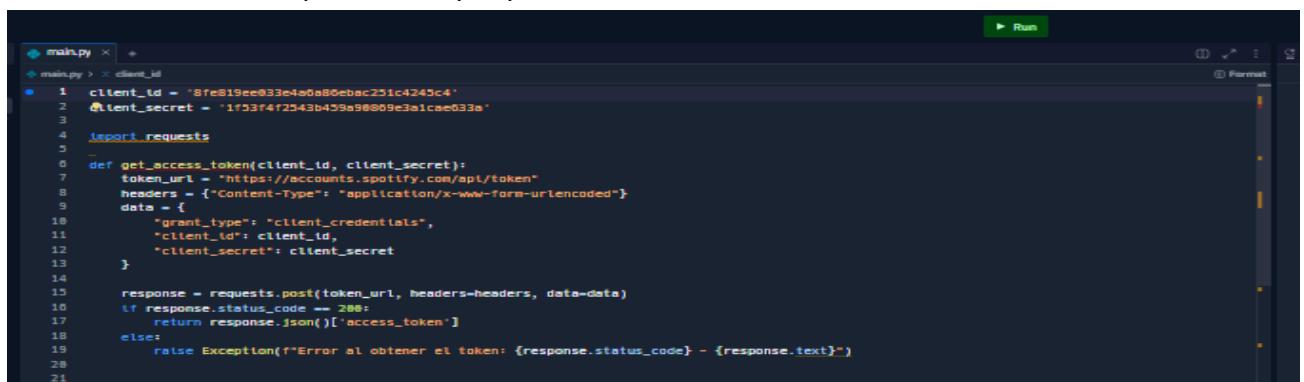
Avanzamos con los últimos detalles y la configuración del Bot de Telegram.

*Por más que por momentos surgían problemas de código, el bot funcionó siempre con la base y fue mejorando por medio de optimizaciones clase a clase.

```
1 import telebot
2 import requests
3
4 # Configura tu bot de Telegram
5 TOKEN =
6     '7571232903:AAF6_CnQviBkr4GWjMcDKxE0w6qa9Rv
7     ajGA'
8     bot = telebot.TeleBot(TOKEN)
9
10    # Funciones de Spotify
11    def get_access_token(client_id,
12        client_secret):
13        token_url =
14            "https://accounts.spotify.com/api/token"
15        headers = {
16            "Content-Type": "application/x-www-
17            form-urlencoded"
18        }
19        data = {
20            "grant_type": "client_credentials",
21            "client_id": client_id,
22            "client_secret": client_secret
23        }
24        response = requests.post(token_url,
25            headers=headers, data=data)
26        if response.status_code == 200:
27            return response.json()
28        [ 'access_token' ]
29    else:
30        raise Exception(f"Error al obtener
```

ENTREGA FINAL:

Creamos un bot de telegram con el objetivo de venderlo a una plataforma de venta de eventos musicales, shows, recitales. Este bot va a solicitarte el nombre del artista o banda, te brindará información certera y optará recomendarte artistas similares. Además te va a enviar su merch oficial para la compra previa al show.



```
main.py > x client_id
main.py > x client_id
1 client_id = '8fe819ee033e4a0a86ebac251c4245c4'
2 client_secret = '1f53f4f2543b459a9009e3a1cae033a'
3
4 import requests
5
6 def get_access_token(client_id, client_secret):
7     token_url = "https://accounts.spotify.com/api/token"
8     headers = {"Content-Type": "application/x-www-form-urlencoded"}
9     data = {
10         "grant_type": "client_credentials",
11         "client_id": client_id,
12         "client_secret": client_secret
13     }
14     response = requests.post(token_url, headers=headers, data=data)
15     if response.status_code == 200:
16         return response.json()['access_token']
17     else:
18         raise Exception(f"Error al obtener el token: {response.status_code} - {response.text}")
19
20
21
22
23
```

```

21
22     try:
23         token = get_access_token(client_id, client_secret)
24         print("Token de acceso:", token)
25     except Exception as e:
26         print(e)
27
28     import telebot
29     import requests
30
31     # Configura tu bot de Telegram
32     TOKEN = '7571232983:AAFb_CnQvBkr4GhjMcDKxE8w0qa9RvaJGA'
33     bot = telebot.TeleBot(TOKEN)
34
35     # Configuración de Spotify
36     client_id = '8fe819ee03e4a6a86ebac231c4245ca'
37     client_secret = '4f0334f229a0439a9000e3a1cae033a'
38
39     # Función para obtener el token de acceso de Spotify
40     def get_access_token(client_id, client_secret):
41         token_url = "https://accounts.spotify.com/api/token"
42         headers = {"Content-Type": "application/x-www-form-urlencoded"}
43         data = {
44             "grant_type": "client_credentials",
45             "client_id": client_id,
46             "client_secret": client_secret
47         }
48         response = requests.post(token_url, headers=headers, data=data)
49         if response.status_code == 200:
50             return response.json()['access_token']
51         else:
52             raise Exception(f"Error al obtener el token: {response.status_code} - {response.text}")
53
54     # Función para obtener el ID de un artista
55     def get_artist_id(access_token, artist_name):
56         search_url = f"https://api.spotify.com/v1/search?q={artist_name}&type=artist"
57         headers = {"Authorization": f"Bearer {access_token}"}
58         response = requests.get(search_url, headers=headers)
59         if response.status_code == 200:
60             artists = response.json().get('artists', {}).get('items', [])
61             if artists:
62                 return artists[0]['id']
63             else:
64                 raise Exception("Artista no encontrado.")
65         else:
66             raise Exception(f"Error al buscar el artista: {response.status_code} - {response.text}")
67
68     # Función para obtener artistas relacionados
69     def get_related_artists(access_token, artist_name):
70         artist_id = get_artist_id(access_token, artist_name)
71         url = f"https://api.spotify.com/v1/artists/{artist_id}/related-artists"
72         headers = {"Authorization": f"Bearer {access_token}"}
73         response = requests.get(url, headers=headers)
74         if response.status_code == 200:
75             return response.json()
76         else:
77             raise Exception(f"Error al obtener artistas relacionados: {response.status_code} - {response.text}")
78
79     # Comando para buscar artistas relacionados
80     @bot.message_handler(commands=['artista'])
81     def handle_artist(message):
82         try:
83             artist_name = message.text.split(maxsplit=1)[1]
84             token = get_access_token(client_id, client_secret)
85             related_artists = get_related_artists(token, artist_name)
86             response_message = "Artistas relacionados:\n"
87             for artist in related_artists['artists']:
88                 response_message += f"- {artist['name']}\n"
89             bot.send_message(message.chat.id, response_message)
90         except Exception as e:
91             bot.send_message(message.chat.id, str(e))
92
93     # Función para buscar productos en Mercado Libre
94     def search_mercado_libre(query):
95         site_id = "MLA"
96         url = f"https://api.mercadolibre.com/sites/{site_id}/search"
97         params = {'q': query}
98         response = requests.get(url, params=params)
99         if response.status_code == 200:
100            return response.json()
101        else:
102            raise Exception(f"Error al buscar en Mercado Libre: {response.status_code} - {response.text}")
103
104     # Comando para buscar productos de un artista en Mercado Libre
105     @bot.message_handler(commands=['buscar'])
106     def handle_search(message):
107         try:
108             query = message.text.split(maxsplit=1)[1]
109             results = search_mercado_libre(query)
110             if 'results' in results and len(results['results']) > 0:
111                 response_message = "Resultados de búsqueda en Mercado Libre:\n"
112                 for item in results['results'][0:5]: # Límite la cantidad de resultados
113                     response_message += f"\nTítulo: {item['title']}\n"
114                     response_message += f"Precio: {item['price']}\n"
115                     response_message += f"Enlace: {item['permalink']}\n\n"
116             bot.send_message(message.chat.id, response_message)
117         else:
118             bot.send_message(message.chat.id, "No se encontraron resultados.")
119         except Exception as e:
120             bot.send_message(message.chat.id, str(e))
121
122     # Comando de bienvenida
123     @bot.message_handler(commands=['start'])
124     def send_welcome(message):
125         bot.send_message(message.chat.id, "¡Bienvenido! Usa /artista <nombre> para obtener artistas relacionados y /buscar <nombre> para buscar productos en Mercado Libre.")
126
127     # Ejecuta el bot
128     if __name__ == "__main__":
129         try:
130             bot.polling()
131         except Exception as e:
132             print(f"Error al iniciar el bot: {e}")

```

El día 27/11 surgió una problemática por el lado de Spotify ya que quitaron la acción de la API para artistas relacionados por lo que tuvimos que optar por utilizar la API de Gemini para realizar esa acción.

```
main.py | +  main.py > ...
1  import telebot
2  import requests
3  from telebot import types
4  import google.generativeai as genai
5  import os
6
7  # Configuración de las claves API
8  os.environ["API_KEY"] = "AIzaSyC1RJq9b0heAcafFGSKhE8vaDgcNT8PmDQ" # Reemplaza con tu
9  API Key de Google
10 genai.configure(api_key=os.environ["API_KEY"])
11
12 # Token del bot de Telegram
13 TOKEN = '7571232983:AAF6_CnQviBkr4GwjMcDKxE@w6qa9RvejGA'
14 bot = telebot.TeleBot(TOKEN)
15
16 # Función para obtener merchandising de un artista desde Mercado Libre (simulación)
17 def get_merch_from_mercadolibre(artist_name):
18     search_url = f"https://listado.mercadolibre.com.ar/{artist_name.replace(' ', '-').lower()}-merchandise"
19     return search_url
20
21 # Función para mostrar la información del artista
22 def show_artist_info(message, artist_name):
23     try:
24         # Generar información básica del artista con IA
25         pregunta = f"¿Qué puedes decirme de {artist_name} como artista, su estilo
26 musical y su estética?"
27         respuesta = genai.GenerativeModel('gemini-1.5-flash-
28 latest').generate_content(pregunta)
29
30         # Componer el mensaje con información generada por IA
31         if respuesta and hasattr(respuesta, 'text'):
32             artist_message = f"**Información sobre
33 {artist_name}:**\n\n{respuesta.text}\n"
34         else:
35             artist_message = f"No se pudo obtener información detallada sobre
36 {artist_name}."
37
38         # Enviar la información del artista al usuario
39         bot.send_message(message.chat.id, artist_message)
40
41         # Obtener el merchandising del artista
42         merch_url = get_merch_from_mercadolibre(artist_name)
43         bot.send_message(message.chat.id, f"¡Aquí está el merchandising disponible
44 para {artist_name} en Mercado Libre! \n{merch_url}")
45
46     except Exception as e:
47         bot.send_message(message.chat.id, f"Error al obtener información del artista:
48 ({e})")
49
50 # Función para mostrar artistas relacionados usando la API de IA
51 def show_related_artists(message, artist_name):
52     try:
53         # Usar la API de IA para generar artistas relacionados
54         pregunta = f"Dame una lista breve de artistas relacionados con {artist_name},
55 con estilos similares o influencias compartidas."
56         respuesta = genai.GenerativeModel('gemini-1.5-flash-
57 latest').generate_content(pregunta)
58
59         if respuesta and hasattr(respuesta, 'text'):
60             # Procesar el texto generado para crear botones de artistas relacionados
61             related_artists = respuesta.text.split("\n")
62             markup = types.InlineKeyboardMarkup()
63
64             for artist in related_artists[:5]: # Limitar a 5 artistas relacionados
65                 artist_clean = artist.strip().replace(" ", "-").lower()
66                 mercado_libre_url =
67                 f"https://listado.mercadolibre.com.ar/{artist_clean}-merchandise"
68                 button = types.InlineKeyboardButton(artist.strip(),
69                 url=mercado_libre_url)
70                 markup.add(button)
71
72             # Enviar el mensaje con los botones
73             bot.send_message(
74                 message.chat.id,
75                 f"Aquí tienes algunos artistas relacionados con {artist_name}.
76 ¡Explora su merchandising!",
77                 reply_markup=markup)
78         else:
79             bot.send_message(message.chat.id, "No pude generar una lista de artistas
80 relacionados. Intenta de nuevo más tarde.")
81
82     except Exception as e:
83         bot.send_message(message.chat.id, f"Error al obtener artistas relacionados:
84 ({e})")
85
86 # Comando para obtener información del artista
87 @bot.message_handler(commands=['artista'])
88 def handle_artist(message):
89     try:
90         artist_name = message.text.split(maxsplit=1)[1]
91         show_artist_info(message, artist_name)
92         show_related_artists(message, artist_name) # Llamamos a la función para
93         # mostrar artistas relacionados
94     except IndexError:
95         bot.send_message(message.chat.id, "Por favor, proporciona el nombre de un
96         artista después del comando /artista.")
97     except Exception as e:
98         bot.send_message(message.chat.id, f"Error al procesar el comando: ({e})")
99
100 # Comando de bienvenida
101 @bot.message_handler(commands=['start'])
102 def send_welcome(message):
103     bot.send_message(message.chat.id, "¡Bienvenido/a! Soy Wanda de Spottel y te ayudo
104     a que encuentres merch para tu próximo show. Usa /artista <nombre> para obtener
105     información.")
106
107 # Responder a cualquier mensaje con la IA
108 @bot.message_handler(func=lambda message: True)
109 def responder_mensaje(message):
110     pregunta = message.text
111     try:
112         respuesta = genai.GenerativeModel('gemini-1.5-flash-
113         latest').generate_content(pregunta)
114         if respuesta and hasattr(respuesta, 'text'):
115             bot.send_message(message.chat.id, respuesta.text)
116         else:
117             bot.send_message(message.chat.id, "No pude generar una respuesta. Intenta
118             de nuevo más tarde.")
119     except Exception as e:
120         bot.send_message(message.chat.id, f"Error al generar la respuesta: ({e})")
121
122 print("Bot de Telegram iniciado")
123 bot.polling()
```