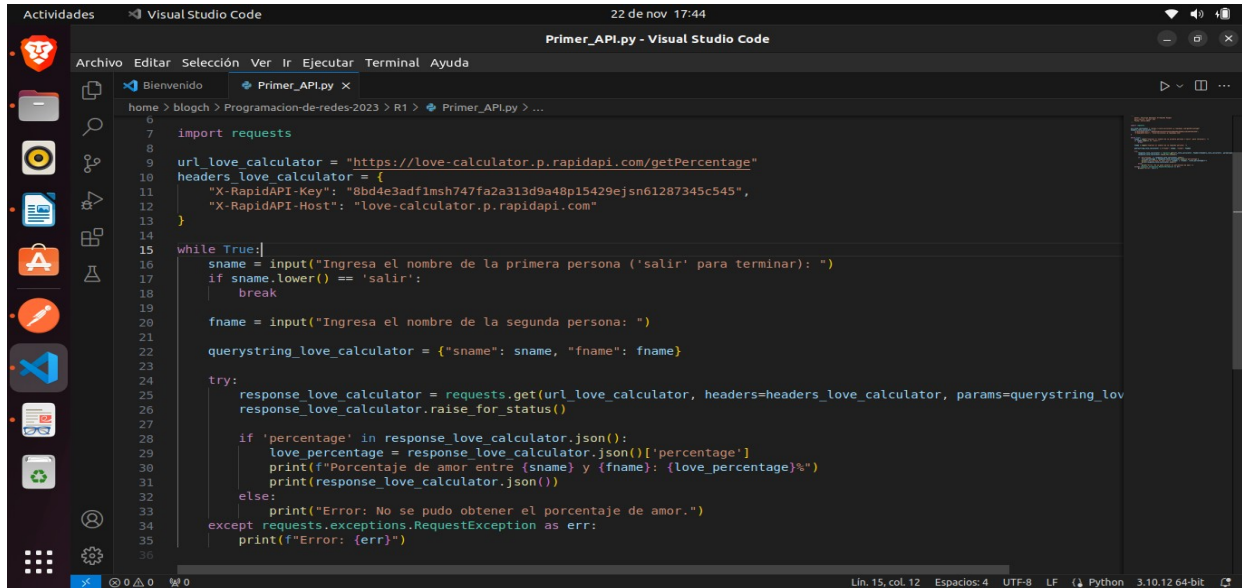


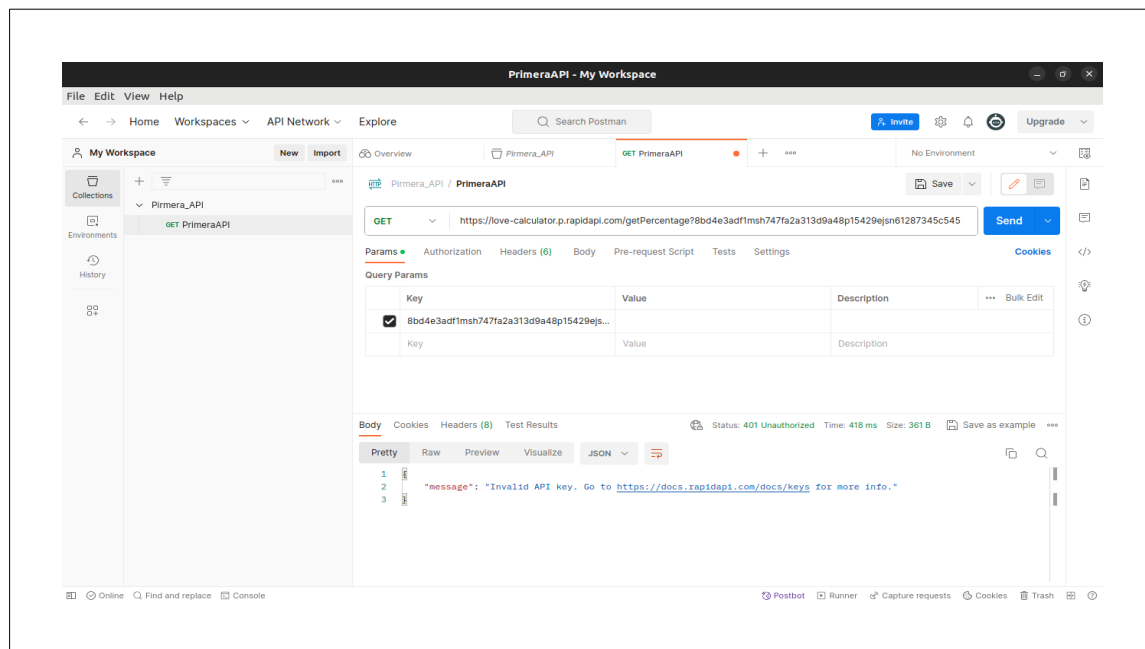
Primera API

Este script utiliza la Love Calculator API para calcular y mostrar el "porcentaje de amor" entre dos personas, según los nombres ingresados. La Love Calculator es un servicio en línea que pretende calcular la compatibilidad amorosa entre dos individuos según sus nombres.



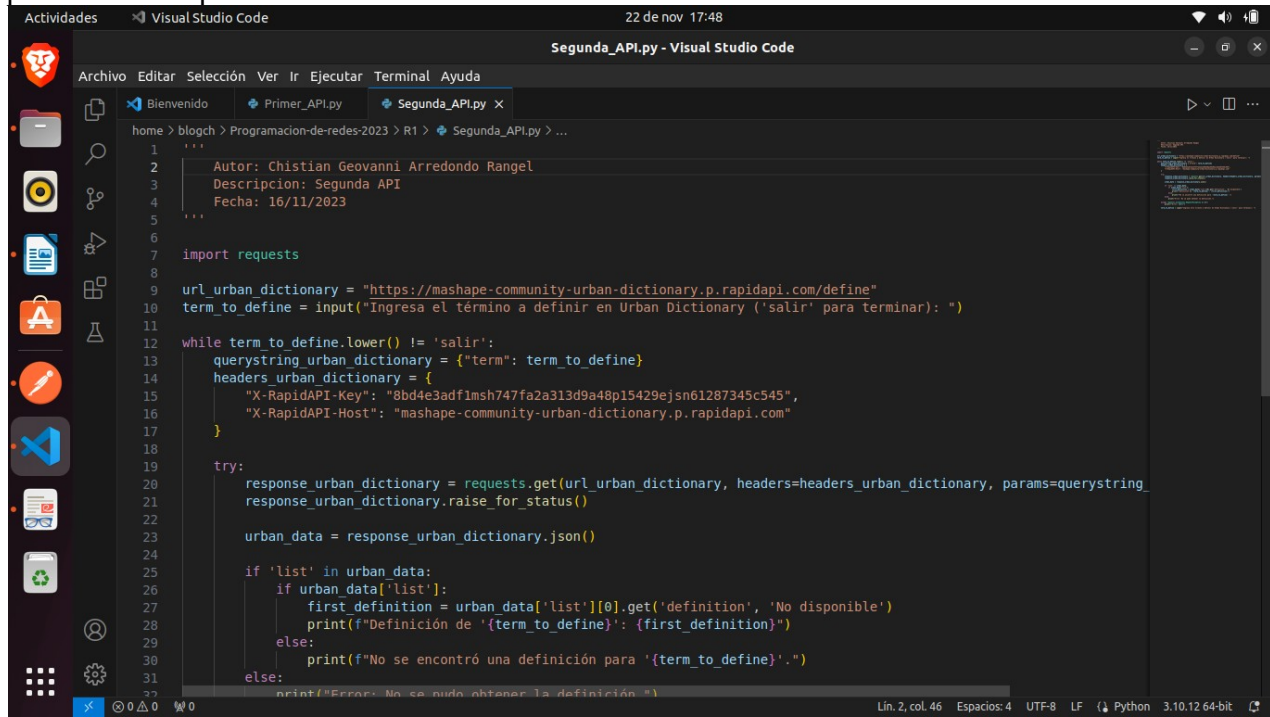
```
6
7 import requests
8
9 url_love_calculator = "https://love-calculator.p.rapidapi.com/getPercentage"
10 headers_love_calculator = {
11     "X-RapidAPI-Key": "8bd4e3adflmsh747fa2a313d9a48p15429ejsn61287345c545",
12     "X-RapidAPI-Host": "love-calculator.p.rapidapi.com"
13 }
14
15 while True:
16     sname = input("Ingresa el nombre de la primera persona ('salir' para terminar): ")
17     if sname.lower() == 'salir':
18         break
19
20     fname = input("Ingresa el nombre de la segunda persona: ")
21
22     querystring_love_calculator = {"sname": sname, "fname": fname}
23
24     try:
25         response_love_calculator = requests.get(url_love_calculator, headers=headers_love_calculator, params=querystring_love_calculator)
26         response_love_calculator.raise_for_status()
27
28         if 'percentage' in response_love_calculator.json():
29             love_percentage = response_love_calculator.json()['percentage']
30             print(f"Porcentaje de amor entre {sname} y {fname}: {love_percentage}%")
31             print(response_love_calculator.json())
32         else:
33             print("Error: No se pudo obtener el porcentaje de amor.")
34     except requests.exceptions.RequestException as err:
35         print(f"Error: {err}")
36
```

Prueba de primera API con Postman:



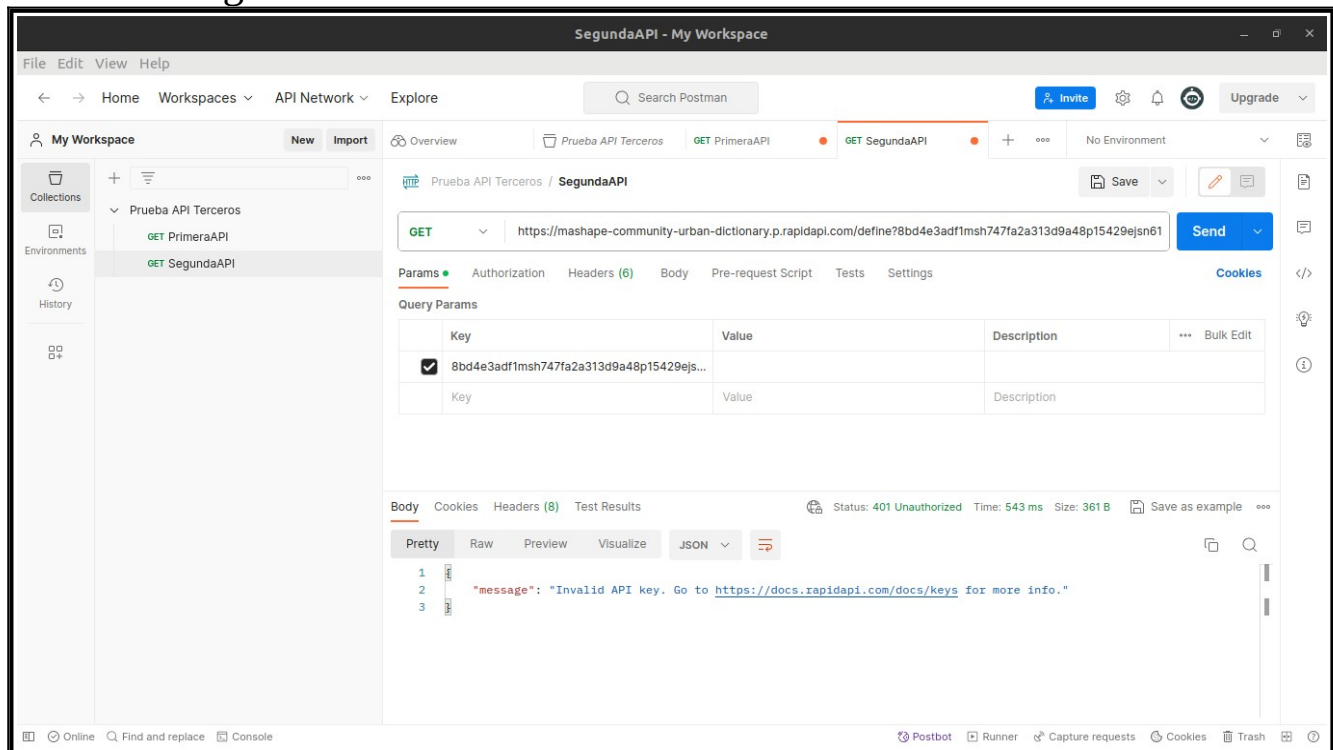
Segunda API

Este script utiliza la Urban Dictionary API para obtener la definición de un término ingresado por el usuario. La Urban Dictionary es un diccionario en línea que contiene definiciones de jergas y expresiones coloquiales utilizadas en la cultura urbana



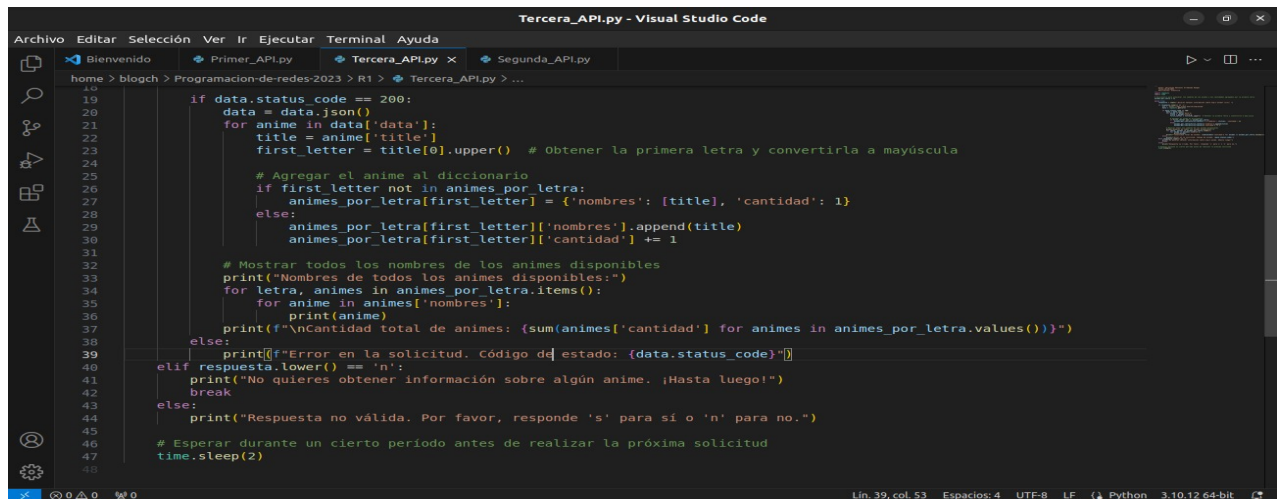
```
1 '''
2 Autor: Chistian Geovanni Arredondo Rangel
3 Descripción: Segunda API
4 Fecha: 16/11/2023
5 '''
6
7 import requests
8
9 url_urban_dictionary = "https://mashape-community-urban-dictionary.p.rapidapi.com/define"
10 term_to_define = input("Ingresa el término a definir en Urban Dictionary ('salir' para terminar): ")
11
12 while term_to_define.lower() != 'salir':
13     querystring_urban_dictionary = {"term": term_to_define}
14     headers_urban_dictionary = {
15         "X-RapidAPI-Key": "8bd4e3adf1msh747fa2a313d9a48p15429ej5n61287345c545",
16         "X-RapidAPI-Host": "mashape-community-urban-dictionary.p.rapidapi.com"
17     }
18
19     try:
20         response_urban_dictionary = requests.get(url_urban_dictionary, headers=headers_urban_dictionary, params=querystring_urban_dictionary)
21         response_urban_dictionary.raise_for_status()
22
23         urban_data = response_urban_dictionary.json()
24
25         if 'list' in urban_data:
26             if urban_data['list']:
27                 first_definition = urban_data['list'][0].get('definition', 'No disponible')
28                 print(f"Definición de '{term_to_define}': {first_definition}")
29             else:
30                 print(f"No se encontró una definición para '{term_to_define}'.")
31         else:
32             print("Error: No se pudo obtener la definición.")
33     except requests.exceptions.HTTPError as http_err:
34         print(f"HTTP error: {http_err}")
35     except requests.exceptions.RequestException as req_err:
36         print(f"RequestException: {req_err}")
37     term_to_define = input("Ingresa el término a definir en Urban Dictionary ('salir' para terminar): ")
38
39 print("Programa terminado.")
```

Prueba de segunda API con Postman:



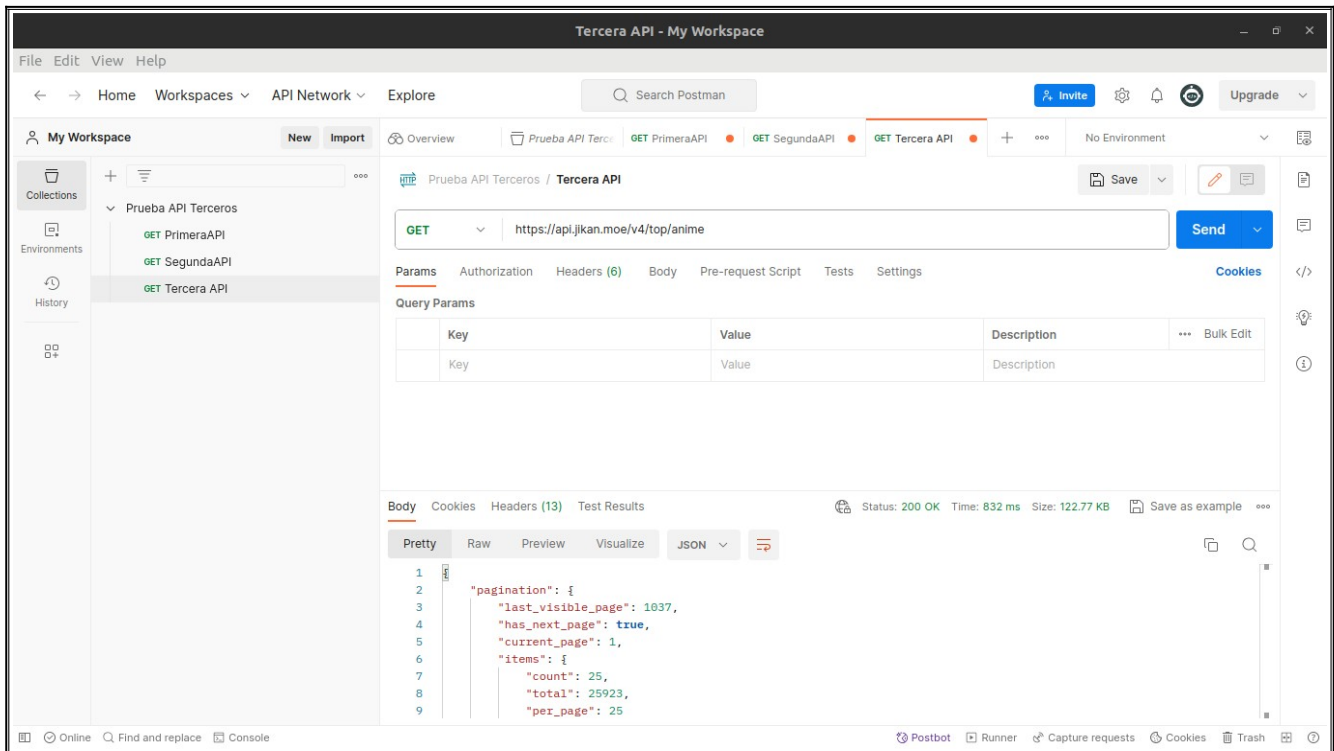
Tercera API

La API Jikan proporciona acceso a datos de MyAnimeList, un sitio web de base de datos de anime y manga. En el código, se utiliza para obtener información sobre los mejores animes, organizándolos por la primera letra de su título. Luego, da la opción al usuario de explorar los nombres de los animes o obtener información detallada sobre uno en particular.



```
19 if data.status_code == 200:
20     data = data.json()
21     for anime in data['data']:
22         title = anime['title']
23         first_letter = title[0].upper() # Obtener la primera letra y convertirla a mayúscula
24
25     # Agregar el anime al diccionario
26     if first_letter not in animes_por_letra:
27         animes_por_letra[first_letter] = {'nombres': [title], 'cantidad': 1}
28     else:
29         animes_por_letra[first_letter]['nombres'].append(title)
30         animes_por_letra[first_letter]['cantidad'] += 1
31
32     # Mostrar todos los nombres de los animes disponibles
33     print("Nombres de todos los animes disponibles:")
34     for letra, animes in animes_por_letra.items():
35         for anime in animes['nombres']:
36             print(anime)
37     print(f"\nCantidad total de animes: {sum(animes['cantidad'] for animes in animes_por_letra.values())}")
38 else:
39     print(f"Error en la solicitud. Código de estado: {data.status_code}")
40 elif respuesta.lower() == 'n':
41     print("No quieres obtener información sobre algún anime. ¡Hasta luego!")
42     break
43 else:
44     print("Respuesta no válida. Por favor, responde 's' para sí o 'n' para no.")
45
46 # Esperar durante un cierto periodo antes de realizar la próxima solicitud
47 time.sleep(2)
```

Prueba de tercera API con Postman:



The screenshot shows the Postman interface with a workspace named "Tercera API - My Workspace". A GET request is configured to the URL `https://api.jikan.moe/v4/top/anime`. The response status is 200 OK, with a time of 832 ms and a size of 122.77 KB. The response body is displayed in JSON format:

```
1 {
2   "pagination": {
3     "last_visible_page": 1037,
4     "has_next_page": true,
5     "current_page": 1,
6     "items": {
7       "count": 25,
8       "total": 25923,
9       "per_page": 25
```