DOCUMENTACION: SEMANA 2

EQUIPO 5: ROTOPLAS HAS LEFT

INTEGRANTES:

Pedro Antonio Soto Cantú 1960073

Martha Iraís Mejía de la Cruz

Héctor Alejandro Hernández Santillán

Diana Monserrat Blanco González 2162825

José Ángel Sánchez Flores

Josué Espinoza López 2064662

1 DE MAYO 2025

PROCESO REALIZADO EN LA SEMANA 2

Solamente de modo de recordatorio, estamos desarrollando una página tipo “WORDLE”, que consiste en adivinar palabras ingresando alguna palabra aleatoriamente y el programa te estará guiando si estás cerca o lejos de obtener la palabra deseada. En este caso, sería similar, pero con Pokémon, escribiendo sus nombres y el programa te entregará diferentes características (tipo(s), hábitat, peso, altura), así sucesivamente hasta que tengas claro qué Pokémon es.

Para la semana dos de nuestro proyecto integrador realizamos una consulta más profunda a la *POKEAPI*, la cual tiene detalladamente cada una de la información que necesitamos a la hora de tratar de adivinar Pokémon.

Después de rescatar la información que necesitábamos, ahora tocaba la extracción de los datos, pero primero utilizamos un código para obtener el nombre del Pokémon para probar la API y además de poder aprender a usarlo.

```python
import requests

url = "https://pokeapi.co/api/v2/pokemon?limit=151"  # Lista de Pokémon, no uno solo
#Nota, modifique el 20 por 151 para que tomara los primeros 151 pokemons (primera generacion)

response = requests.get(url)
data = response.json()

# Verifica qué hay dentro del JSON
print(data.keys())  # Esto debe mostrar: dict_keys(['count', 'next', 'previous', 'results'])

# Ahora accede al primer Pokémon
pokemon = data["results"][0]["name"]

print(pokemon)
```

A esta manera le llamamos, “Mi primera vez”.

Utilizamos como herramienta, YouTube (Ya que ahí viene de todo) y pudimos guiarnos para realizar el código de una buena manera y funcional. Limitamos el número de Pokémon a 151, ya que como son una fumada de Pokémon, sólo aplicaremos la primera generación (que es mítica).

Ya con el código de prueba en nuestras manos, ahora tocaba hacer la técnica de limpieza para obtener los datos de cada uno de los Pokémon, primeramente, investigamos la manera más adecuada e intentamos como base utilizar el código anterior mostrado pero esta vez añadiendo el valor de x para seleccionar un Pokémon dependiendo su número en la Pokédex.

En nuestra segunda versión del código, tratamos de incorporar ahora si lo que necesitamos para nuestra página, es decir, para poder extraer los datos pusimos cuatro como manera de prueba para visualizar el funcionamiento de este. Cambiamos la mayoría para una mejor compresión para el programa y así obtener de manera más eficaz los datos.

```
import requests
import random
base_url = "https://pokeapi.co/api/v2/"

def get_pokemon_info(id):
    url = f"{base_url}/pokemon/{id}"
    response = requests.get(url)

    if response.status_code == 200:
        pokemon_data = response.json()
        return pokemon_data
    else:
        print(f"Failed to retrieve data {response.status_code}")
x = random.randrange(1, 151)
pokemon_id = x
pokemon_info = get_pokemon_info(pokemon_id)

if pokemon_info:
    print(f"Name: {pokemon_info["name"].capitalize()}")
    print(f"Id: {pokemon_info["id"]}")
    print(f"Altura: {pokemon_info["height"]}", "m")
    print(f"Peso: {pokemon_info["weight"]}", "Kg")
```

A través de diferentes ajustes al código pudimos finalmente crear una versión estable que pueda brindar los datos que necesitamos para la estructuración de los mismos datos.

Al realizar diferentes pruebas pudimos reconstruir el código para que pudieran entrar los diccionarios y ahora nos brinden más datos de cada Pokémon. A pesar de que los tipos puedan estar todavía en una fase temprana, los demás datos están listos para poder extraerse.

```python
import requests
import json
base_url = "https://pokeapi.co/api/v2/"
x=0
pokemon = {}
while x < 151:
    def get_pokemon_info(id): #Una funcion para usar después
        url = f"{base_url}/pokemon/{id}" #Esto para que podamos cambiar la url a lo que necesitamos, una url que puede cambiar la id
        response = requests.get(url)

        if response.status_code == 200: #Esto es por si la URL esta caida o tiene problemas no tire directamente error
            pokemon_data = response.json()
            return pokemon_data
        else:
            print(f"Fallo al recuperar la información {response.status_code}") #Aquí terminaria el programa si no logra conectar con la URL
    x = x+1 #elige un numero del 1 al 151
    pokemon_id = x #lo graba en la id
    pokemon_info = get_pokemon_info(pokemon_id) #saca la informacion de la id
    y = f"{pokemon_info["name"].capitalize()}"
    if pokemon_info: #Ando pensando en quitar este if, pero no estoy seguro
        Tipos = [typ["type"]["name"] for typ in pokemon_info["types"]] #este tuvo que ser diferente, porque al haber 2 instancias de "Types" soltaba error
        pokemon.update({y:  "id: " f"{pokemon_info["id"]}, " "altura: " f"{pokemon_info["height"]}" "m, " "Peso: " f"{pokemon_info["weight"]}" "Kg, " "Pokemon de tipo: " })
json_string = json.dumps(pokemon, indent=4)
print(json_string)
with open("datos estructurados.json", "w") as f:
    json.dump(pokemon, f, indent=4)
```

Finalmente, al realizar el “extractor” de los datos ahora si podremos tener la información de los 151 Pokémon, los cuales solamente el hábitat y el tipo faltarían, pero por diferentes razones, ya sea por que no sabemos cómo o por alguna incoherencia al ingresarlo en el código.


```json
{
    "Bulbasaur": "id: 1, altura: 7m, Peso: 69Kg, Pokemon de tipo: ",
    "Ivysaur": "id: 2, altura: 10m, Peso: 130Kg, Pokemon de tipo: ",
    "Venusaur": "id: 3, altura: 20m, Peso: 1000Kg, Pokemon de tipo: ",
    "Charmander": "id: 4, altura: 6m, Peso: 85Kg, Pokemon de tipo: ",
    "Charmeleon": "id: 5, altura: 11m, Peso: 190Kg, Pokemon de tipo: ",
    "Charizard": "id: 6, altura: 17m, Peso: 905Kg, Pokemon de tipo: ",
    "Squirtle": "id: 7, altura: 5m, Peso: 90Kg, Pokemon de tipo: ",
    "Wartortle": "id: 8, altura: 10m, Peso: 225Kg, Pokemon de tipo: ",
    "Blastoise": "id: 9, altura: 16m, Peso: 855Kg, Pokemon de tipo: ",
    "Caterpie": "id: 10, altura: 3m, Peso: 29Kg, Pokemon de tipo: ",
    "Metapod": "id: 11, altura: 7m, Peso: 99Kg, Pokemon de tipo: ",
    "Butterfree": "id: 12, altura: 11m, Peso: 320Kg, Pokemon de tipo: ",
    "Weedle": "id: 13, altura: 3m, Peso: 32Kg, Pokemon de tipo: ",
    "Kakuna": "id: 14, altura: 6m, Peso: 100Kg, Pokemon de tipo: ",
    "Beedrill": "id: 15, altura: 10m, Peso: 295Kg, Pokemon de tipo: ",
    "Pidgey": "id: 16, altura: 3m, Peso: 18Kg, Pokemon de tipo: ",
    "Pidgeotto": "id: 17, altura: 11m, Peso: 300Kg, Pokemon de tipo: ",
    "Pidgeot": "id: 18, altura: 15m, Peso: 395Kg, Pokemon de tipo: ",
    "Rattata": "id: 19, altura: 3m, Peso: 35Kg, Pokemon de tipo: ",
    "Raticate": "id: 20, altura: 7m, Peso: 185Kg, Pokemon de tipo: ",
    "Spearow": "id: 21, altura: 3m, Peso: 20Kg, Pokemon de tipo: ",
    "Fearow": "id: 22, altura: 12m, Peso: 380Kg, Pokemon de tipo: ",
    "Ekans": "id: 23, altura: 20m, Peso: 69Kg, Pokemon de tipo: ",
    "Arbok": "id: 24, altura: 35m, Peso: 650Kg, Pokemon de tipo: ",
    "Pikachu": "id: 25, altura: 4m, Peso: 60Kg, Pokemon de tipo: ",
    "Raichu": "id: 26, altura: 8m, Peso: 300Kg, Pokemon de tipo: ",
    "Sandshrew": "id: 27, altura: 6m, Peso: 120Kg, Pokemon de tipo: ",
    "Sandslash": "id: 28, altura: 10m, Peso: 295Kg, Pokemon de tipo: ",
    "Nidoran-f": "id: 29, altura: 4m, Peso: 70Kg, Pokemon de tipo: ",
    "Nidorina": "id: 30, altura: 8m, Peso: 200Kg, Pokemon de tipo: ",
    "Nidoqueen": "id: 31, altura: 13m, Peso: 600Kg, Pokemon de tipo: ",
```


- Esta imagen muestra unos cuantos Pokémon, pero en GitHub está toda la generación.