UANL

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FCFM

PROGRAMACION BASICA

OSVALDO HABIB GONZÁLEZ GONZÁLEZ

PROYECTO FINAL PIA

SEBASTIAN CALDERON CARRILLO *2087472*

HECTOR ALAN HERNANDEZ GONZALEZ *2082913*

LEONARDO ISAAC VELA CORTES *2154477*

Planteamiento del problema:

Que fenómeno o situación a analizar.

Se busca analizar las diferentes estadísticas de los pokémons base. Estadísticas como ataque, defensa, vida, velocidad, altura y peso.

Porque es relevante

Existen personas que dedican mucho tiempo a competir en cosas relacionadas con Pokémon, por lo que les y nos es de ayuda para analizar cuáles serían nuestras mejores opciones para utilizar contra algún rival.

Datos que se necesitan obtener

La API que hemos escogido cuenta con un sinfín de datos, pero en nuestro caso los que más nos son útiles son los datos numéricos, en los cuales se incluyen ataque, defensa, vida, velocidad, altura y peso. Estos datos intuitivamente van relacionados al nombre de cada Pokémon, siendo este su identificador.

Descripción de la API elegida

La API que escogimos es pública y gratuita. Esta API se creó con la idea de unificar el sitio de donde las demás páginas de análisis Pokémon obtenían su información, volviéndose ya casi un standard en cuanto a datos actualizados se trata. La API tiene una restricción de peticiones HTTPS por hora, la cual puede variar dependiendo del tráfico, pero en general se recomienda usar sabiamente y con cautela la cantidad de peticiones solicitadas. Es por eso por lo que optamos por crear un cache con la información obtenida de la API, y solamente hacemos requests HTTPS cuando necesitemos actualizar la información.

Estructura de datos

Dependiendo de las distintas funciones que hicimos, es que se creó una estructura de datos adecuada para la necesidad a resolver. Aunque, también podemos decir que la estructura “principal” es la que ya nos ofrece la API en sus respuestas, la cual vendría siendo un diccionario iterable con elementos key:value, el key siempre es un string describiendo que es lo que contiene su correspondiente value. El value puede cambiar dependiendo de que es la información que contenga. A veces son listas de diccionarios, listas de tuplas, a veces solo es un integer, etc.

A continuación, describiremos las estructuras de datos utilizadas en las funciones correspondientes:

Función “extraer_names_de_endpoint_a_list” (Utiliza una lista de strings):

[‘name1’, ‘name2’, ‘name3’, ‘name4’]

Función “extraer_informacion_json_a_dict” (Utiliza un diccionario que basicamente es el .json del cual se ingrese el argumento), ejemplo:

```
{
    "abilities": [
        ‘Golpear’,
        ‘Saltar’,
        ‘Correr’
    ],
    ‘height’:185,
    ‘stats’: {
        ‘hp’:80,
        ‘speed’:90.
        ‘attack’:45
    }
}
```

Función “extraer_stats_pokemon_a_dict” (Diccionario con valores stat:value):

```
{
    ‘hp’: 90,
    ‘attack’: 80,
    ‘defense’: 50,
    ‘speed’: 35
}
```

Función “exportar_excel_info_pokemones” (Lista de str y una lista de listas de floats y str) (Cada lista dentro de la lista “Filas” Corresponde a una sola fila)

Columnas = ['Name','height','weight','hp','attack','defense','special-attack','special-defense','speed']

Filas = [[‘bulbasaur’, 7, 69, 45, 49, 49, 65, 65, 45], [‘ivysaur’, 10, 130, 60, 62, 63, 80, 80, 60], [‘venusaur’, 20, 1000, 80, 82, 83, 100, 100, 80]]

Minutas de trabajo

Minuta 2 de mayo de 2025

Roles asignados:

- **Líder:** Sebastián
- **Programador**: Sebastián
- **Diseñador de algoritmo**: Sebastián y Héctor
- **Encargado del diagrama**: Héctor
- **Minutas:** Héctor

Objetivo para el día:

Ponernos de acuerdo en cuanto a la idea del proyecto a realizar, junto con la organización de cada miembro para cumplir los roles requeridos

Acuerdos tomados:

Los roles asignados, la persona que supervisará el trabajo y las actividades de cada uno.

Dificultades encontradas:

Organización para las actividades a realizarse dado que se inició antes de vacaciones.

Próximos pasos:

- Realizar el diagrama de flujo y diseño del algoritmo.
- Realizar la codificación en Python.

Cuadro comparativo de APIs exploradas

Nombre de la API	Descripción	Problemas encontrados
PokeAPI	Es una API que busca unificar y actualizar la información relacionada con Pokémon, esto con el objetivo de que de ella dependan todas las demás páginas que utilicen algún tipo de analítica o datos relacionados a Pokémon.	El único problema que hallamos fue que estan limitadas las peticiones por hora. Aunque no hay una cantidad especifica, solo recomiendan el uso moderado. Esto lo resolvimos mediante utilizando cache
Weatherstack	Cuenta con información actualizada y mantiene un log histórico de datos meteorológicos de todo el mundo	Tiene restricciones de peticiones, así como de funcionabilidad. Por lo que, de manera gratuita solo hubiéramos podido acceder a la temperatura actual, lo cual no nos hubiera servido para hacer análisis estadísticos.
One Call API 3.0	De igual manera que weatherstack, tiene información meteorológica actual e histórica, así como demás utilidades como reconocimiento de texto para analizar la ciudad que se ingresa, aunque tenga tipografías	De manera similar a Weatherstack, muchas funcionalidades estaban bloqueadas, aunque para desbloquearlas el costo era mínimo, optamos por mejor utilizar pokeAPI

Algoritmo

Función actualizar_jsons_de_pokemones(api)

1. Hacer una solicitud GET a (api + 'Pokémon/?limit=100')
2. Si la respuesta es exitosa (status_code = 200):
    a. Guardar la respuesta JSON en el archivo 'cache_pokemon.json'
3. Sino:
    a. Imprimir mensaje de error con el status_code

4. Leer 'cache_pokemon.json'
5. Para cada pokemon en los resultados:
    a. Obtener la URL del pokemon

    b. Hacer GET a la URL y guardar el JSON en 'cache\<nombre_pokemon>.json'

Función extraer_names_de_endpoint_a_list(endpoint)

1. Inicializar lista vacía names
2. Leer archivo 'cache_<endpoint>.json'
3. Parsear contenido JSON
4. Para cada elemento en 'results':
    a. Agregar el campo 'name' a la lista names
5. Retornar names

Función extraer_stats_pokemon_a_dict(nombre)

1. Inicializar diccionario vacío
2. Obtener lista de estadísticas del pokemon (usando extraer_informacion_json_a_dict)
3. Para cada estadística en la lista:
    a. Agregar el nombre de la estadística y su valor al diccionario
4. Retornar el diccionario

Función crear_grafica_de_barras_de_atributo_pokemones(pokemones, atributo)

1. Inicializar diccionario vacío 'stats'
2. Para cada pokemon en la lista:
    a. Obtener el valor del atributo usando extraer_stats_pokemon_a_dict
    b. Agregarlo al diccionario stats
3. Crear listas x e y desde los datos de stats
4. Configurar gráfico de barras usando matplotlib
    a. Título: Atributo de los Pokemones
    b. Etiquetas de barras con los nombres capitalizados
    c. Etiqueta del eje Y con el nombre del atributo
5. Guardar gráfico como PDF
6. Mostrar gráfico

Función crear_grafica_de_barras_de_alturas_pokemones(pokemones)

1. Inicializar diccionario vacío 'stats'
2. Para cada Pokémon:
    a. Obtener altura desde el JSON y multiplicar por 10
    b. Agregar al diccionario stats

3. Crear listas x e y
4. Configurar gráfico de barras con color verde y etiqueta en cm
5. Guardar gráfico como PDF
6. Mostrar gráfico

Función crear_grafica_de_barras_de_pesos_pokemones(pokemones)

1. Inicializar diccionario vacío 'stats'
2. Para cada pokemon:
    a. Obtener peso desde el JSON y dividir entre 10
    b. Agregar al diccionario stats
3. Crear listas x e y
4. Configurar gráfico de barras con color amarillo y etiqueta en kg
5. Guardar gráfico como PDF
6. Mostrar gráfico

Función exportar_excel_info_pokemones(pokemones)

1. Inicializar lista vacía data
2. Definir columnas del Excel
3. Para cada pokemon:
    a. Crear lista temporal con nombre, altura y peso
    b. Agregar cada estadística (hp, attack, etc.)
    c. Agregar lista temporal a data
4. Crear DataFrame con pandas
5. Exportar a archivo Excel

Función: obtener_media(pokemones, stat)

1. Inicializar lista vacía
2. Para cada pokemon:
    a. Obtener valor del stat y agregarlo a la lista
3. Calcular la media: suma de valores / cantidad
4. Imprimir y retornar la media

Función obtener_mediana(pokemones, stat)

1. Inicializar lista vacía
2. Para cada pokemon:
    a. Obtener valor del stat y agregarlo a la lista
3. Ordenar la lista

4. Si la longitud es impar:
    a. La mediana es el valor central
5. Si es par:
    a. La mediana es el promedio de los dos valores centrales
6. Imprimir y retornar la mediana

Función obtener_moda(pokemones, stat)

1. Inicializar lista vacía
2. Para cada pokemon:
3. Obtener valor del stat y agregarlo a la lista
4. Crear diccionario de frecuencia
5. Para cada valor en la lista:
    a. Contar repeticiones en el diccionario
6. Buscar el valor con mayor frecuencia
7. Imprimir y retornar la moda

Resumen del proyecto con hallazgos relevantes

A través de la creación de este proyecto se necesitó mucha investigación en cuanto a las funcionalidades de las librerías utilizadas, así como la funcionalidad de la API. Comprendimos la importancia de la reutilización de codigo, o sea, la programación modular, ya que es más fácil de dar mantenimiento, de entender, de compactar el codigo y en general ser más fácil.

Para el control de versiones, se tuvo que investigar el funcionamiento de Git, y a su vez, publicarlo en GitHub, lo cual nos facilitó crear backups de nuestro codigo y poder regresar a una función estable o implementar nuevas cosas de manera más fácil.

Guion del podcast

Buen día, mi nombre es Sebastian Calderon. Nuestro equipo creo un programa que buscaba satisfacer la necesidad de analizar datos referentes a la saga Pokémon. Fue así que decidimos utilizar la página PokeAPI, la cual provee de los datos más actualizados de todo lo relacionado con los videojuegos de la saga. Decidimos solamente utilizar datos del tipo estadísticas iniciales de los pokemones, como el ataque, la defensa, la vida, la velocidad, entre otros. Pero, cabe destacar que la API proporciona más datos como las estadísticas de todos los ítems del juego como berries, potenciadores, curaciones, etc.

Dividimos nuestro programa en dos archivos: Uno en el que se definen todas las funciones disponibles para utilizar y otro que contendría la lógica del programa utilizando la importación del documento que contiene las funciones.

El documento que contiene las funciones tiene 11 modulos que en algunos casos de utilizan entre sí. A continuación, les resumo un poco de todas las funciones que implementamos:

Actualizar_jsons_de_pokemons es una función que actualiza todos los jsons, los cuales se utilizan como cache para no tener que hacer peticiones a la API cada que necesitemos obtener información. También nos sirve en caso de que la API se caiga, ya que en sí, el programa utiliza información localmente.

La función extraer_names_de_endpoint_a_lista es una función que sirve para de cierta manera, recorrer la estructura que tiene la API, ya que la primera request que se puede hacer retorna un listado por ejemplo de todos los nombres de los pokemones. Posteriormente es cuando se puede acceder individualmente a los datos de cada uno de los pokemones

La función extraer_informacion_json_a_dict funciona para traducir el json obtenido de alguna url a un diccionario de Python, el cual ya sabemos cómo recorrer, leer, etc. Esta es de las funciones más reutilizadas, ya que es básico para el manejo de la información obtenida.

La función extraer_stats_pokemon_a_dict(nombre) accede específicamente a los stats de un pokemon y los retorna en forma de diccionario de Python. Este contiene datos como ataque, defensa, velocidad, vida, etc.

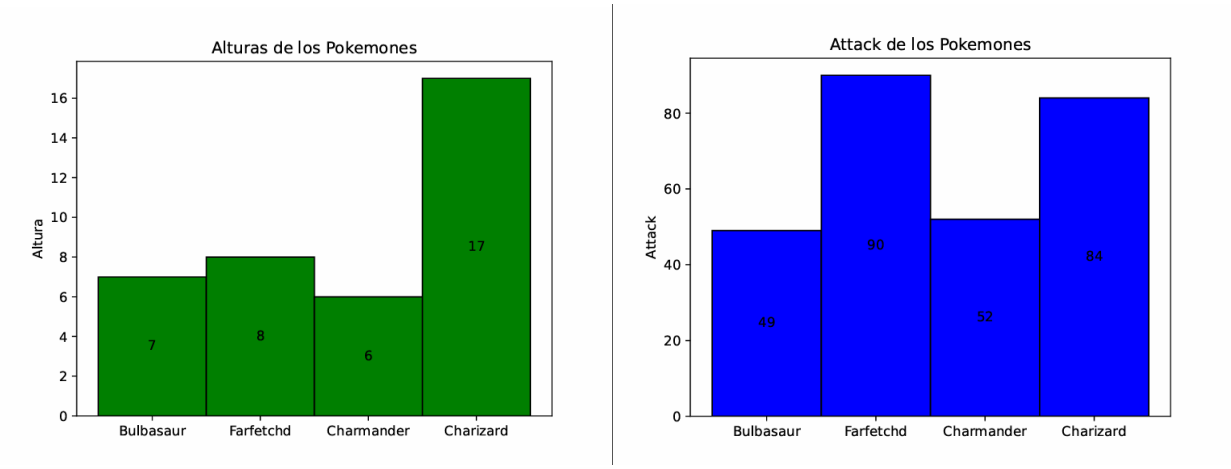
Tambien existen 3 funciones parecidas para crear graficas de barras que comparan los datos de distintos pokemones. Cabe destacar que, al mismo tiempo, también exporta la gráfica en pdfs para poder utilizarla en más documentos.

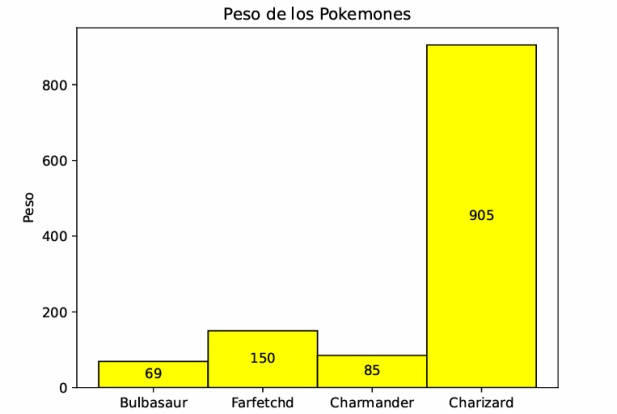
Asi como se exportan las gráficas a pdfs, existe una función que exporta un Excel que contiene todos los datos de los pokemones ingresadaos

Por último, tenemos 3 funciones para datos estadísticos de tendencia central, o sea, moda, mediana y moda. Las cuales imprimen y retornan el valor especificado para poder asignárselo a alguna variable si así se requiere

Capturas de pantalla y elementos visuales de apoyo

En la carpeta programa se incluyen las exportaciones de pdfs referentes a las gráficas que se realizaron, así como el Excel exportado. Aquí también se incluyen:





	Name	height	weight	hp	attack	defense	pecial-attac	ecial-defen	speed
0	bulbasaur	7	69	45	49	49	65	65	45
1	ivysaur	10	130	60	62	63	80	80	60
2	venusaur	20	1000	80	82	83	100	100	80
3	charmand	6	85	39	52	43	60	50	65
4	charmeleo	11	190	58	64	58	80	65	80
5	charizard	17	905	78	84	78	109	85	100
6	squirtle	5	90	44	48	65	50	64	43
7	wartortle	10	225	59	63	80	65	80	58
8	blastoise	16	855	79	83	100	85	105	78
9	caterpie	3	29	45	30	35	20	20	45
10	metapod	7	99	50	20	55	25	25	30
11	butterfree	11	320	60	45	50	90	80	70
12	weedle	3	32	40	35	30	20	20	50
13	kakuna	6	100	45	25	50	25	25	35
14	beedrill	10	295	65	90	40	45	80	75