DOCUMENTACION: SEMANA 3

EQUIPO 5: ROTOPLAS HAS LEFT

INTEGRANTES:

Pedro Antonio Soto Cantú 1960073

Martha Iraís Mejía de la Cruz 2087489

Héctor Alejandro Hernández Santillán 2069405

Diana Monserrat Blanco González 2162825

José Ángel Sánchez Flores 2225398

Josué Espinoza López 2064662

16 DE MAYO 2025

PROCESO REALIZADO EN LA SEMANA 3

Para esta semana número 3 de nuestro proyecto integrador, el cual es un “POKEDLE” que hace referencia a los programas de adivinar palabras (en este caso, pokemon). Utilizamos el POKEAPI, como ya se ha planteado, con este api estuvimos trabajando durante las semanas hasta conseguir un código sólido. Ahora lo que se realizará es un segundo script que permita leer datos del primer código.

Cabe recalcar que cambiamos la idea original del estilo de adivinar palabras tipo WORLDE a un tipo minijuego clásico llamado “El Ahorcado” que se nos hizo más sencillo para programar, ya que estaba complicado realizar lo demás hecho, pero seguimos el mismo tema, solamente cambiamos la manera de jugarlo.

Para nuestro análisis estadístico, para elegir un pokemon utilizamos un random number generator que es contenido de numpy, es decir, a partir de obtener un número del generador, dependiendo del número obtendremos el pokemon al cual queremos adivinar. Además de que nos dará información de dicho pokemon seleccionado por el programa.

Lo que ocurrió aquí es que para poder guardar los intentos realizados en el programa tuvimos varios problemas, como ciertos errores que daban en algunas filas del código, como el desconocimiento de diferentes métodos para poder aplicar una mejor aplicación del código (cabe destacar que también para que se viese decente).

LECTURA DE DATOS:

¿Qué son los Pokémon? Los Pokémon son criaturas ficticias, que forman parte de una popular franquicia de medios que incluye videojuegos, series de televisión, películas y juegos de cartas. Los Pokémon son capturados y entrenados por entrenadores, y se enfrentan en batallas con otros Pokémon para alcanzar la victoria.

A todo esto ha generado una gran popularidad, así que hemos capturado información sobre estos enfocándonos en la primera generación de Kanto o primera generación en donde nuestro código abre y lee un archivo llamado ´datos_estructurados.json usando la función json.load(). Para crear un diccionario llamado ¨datos¨ que va a darle base a nuestro juego para mostrar pistas y verificar si el nombre del Pokémon es adivinado correctamente. Este archivo contiene información sobre 151 Pokémon ya sea: nombre, tipo, hábitat, color, evolución, etc. Para que le sea más fácil al jugador poder acertar además de agregarle dificultad por la cantidad de variedad de estos.

VALIDACIÓN

La validación de la entrada del jugador es un paso fundamental para garantizar una experiencia fluida y libre de errores. Para ello, se emplea la función "letra_valida(letra)", cuyo propósito es verificar si la letra ingresada por el jugador cumple con ciertos criterios esenciales.

En primer lugar, la función comprueba que la entrada corresponda a una única letra, evitando así caracteres adicionales que puedan generar confusión o resultados inesperados, asimismo, se valida que la letra ingresada forme parte del abecedario permitido en el juego, asegurando que solo se consideren caracteres válidos dentro del contexto establecido.

Otro aspecto crucial de la validación es verificar que la letra no haya sido previamente utilizada, ya sea acertada o fallada en intentos anteriores. Esto previene la repetición de elecciones y garantiza que el juego se desarrolle de manera justa, sin errores que puedan afectar la dinámica o el registro de intentos.

En conjunto, esta validación contribuye a mantener una experiencia de usuario óptima, reduciendo frustraciones y asegurando que cada movimiento realizado por el jugador sea efectivo y conforme a las reglas establecidas.

ANÁLISIS CON NUMPY

En el script con el nombre “Ahorcado.py” hacemos uso de la librería NumPy. Esta librería se usa para generar números enteros de forma aleatoria con un intervalo de 1 a 150. Si bien es un proceso que también podríamos realizar con la librería “Random”. Resulta como para nosotros, ya que coincidentemente hay una cantidad muy cercana de Pokemon en la generación que estamos trabajando.

El intervalo original de NumPy es de 1 a 150. Sin embargo, para poder realizar correctamente la búsqueda de todos los Pokemon de la primera generación, Modificamos el intervalo hasta 151. Antes de esto, es importante instalar la librería NumPy directamente en la PC mediante la consola. Una vez hecho, importar la librería a nuestro script.

Una vez se ingresan los datos que pide el script, en las líneas 11 de nuestro código, NumPy genera un número aleatorio que será convertido a texto. Así nos servirá para realizar la búsqueda de los datos de ese Pokemon, desde el diccionario “datos_estructurados.json” que es el archivo donde tenemos el diccionario con los datos de los Pokémon de la primera generación.

En nuestros otros scripts no tenemos incluida dicha librería por lo que solo usamos la librería para generar el número aleatorio de Pokemon que el usuario deberá de adivinar.

APLICACIÓN DE LOS DATOS

La forma en la que utilizamos los datos de la PokeAPI fue algo diferente a como originalmente teníamos pensado ya que el concepto original tuvo un cambio en esta

Semana_3 ya que se optó por realizar un ahorcado en donde habría que adivinar el nombre del pokemon con una cantidad de intentos limitados.

Para poder integrar los datos en el proyecto utilizamos varias funciones y estructuras específicas que facilitan la interacción con los atributos del Pokemon, ya que al iniciar el Ahorcado, se selecciona de forma aleatoria un Pokemon con la función np.random.randint(1,151), lo que nos daba un numero al azar de ese rango establecido y que fue seleccionado específicamente para solo abarcar la primera generación de pokemon en un diccionario de datos donde se almacenaron los datos extraídos de la PokeAPI.

El nombre del Pokémon se guarda en la variable palabra como una lista de caracteres. Luego se creaba una lista oculta con guiones bajos ('_') de la misma longitud, que representa las letras ocultas que el jugador debe descubrir. Para mostrar el estado actual de la palabra, implementé la función estado(), que imprime los intentos restantes, las letras descartadas y la palabra oculta con las letras acertadas.