

**“Universidad Autónoma de Coahuila”**

“Facultad de sistemas”

Materia: “Calidad y pruebas de software”

“Proyecto Final”

“Pokemon API”

Maestro: Carlos Nassif

Equipo: Joe Marco Zuñiga Guerrero

Salvador Paniagua de la Fuente

Edson Armijo Ruiz

Juan Abimael Camacho Canizales

Introducción:

El proyecto consiste en una aplicación de línea de comandos que permite interactuar con la API de PokéAPI y una base de datos para obtener información y administrar equipos de Pokémon. El programa ha sido desarrollado en Python y utiliza diversas bibliotecas y módulos para realizar solicitudes HTTP, manipular datos en formato JSON y gestionar la conexión con la base de datos.

La aplicación ofrece varias funcionalidades principales, como la búsqueda de información de Pokémon por su nombre o número en la Pokédex, la visualización de los Pokémon de un equipo, la eliminación de Pokémon de un equipo y la capacidad de guardar Pokémon en un equipo específico. Para lograr esto, se ha implementado una serie de funciones y una estructura de menú interactivo que permite al usuario seleccionar las acciones deseadas.

Además, el programa utiliza una base de datos denominada "pokedb" para almacenar los equipos de Pokémon. Se ha creado una tabla específica para este propósito y se emplean funciones de base de datos para realizar operaciones como guardar y eliminar Pokémon de un equipo.

Descripción del Proyecto:

1. Descripción: El programa es una aplicación de línea de comandos que interactúa con la API de PokéAPI y una base de datos para realizar diversas acciones relacionadas con los Pokémon. Permite obtener información de un Pokémon por su nombre o número en la Pokédex, mostrar los Pokémon de un equipo, quitar un Pokémon del equipo y salir del programa.
2. Dependencias: El programa utiliza las siguientes bibliotecas de Python:
   * **requests**: Utilizada para realizar solicitudes HTTP a la API de PokéAPI y obtener datos en formato JSON.
   * **json**: Utilizada para trabajar con datos JSON.
   * **database**: Importa funciones relacionadas con la conexión a la base de datos, creación de tablas y operaciones de guardado y eliminación de Pokémon.
3. Funciones: a. **get\_pokemon\_data(pokemon\_id)**: Esta función recibe un ID o nombre de Pokémon como argumento y devuelve un diccionario con la información del Pokémon, incluyendo número, nombre, movimientos y debilidades. Utiliza la API de PokéAPI para obtener los datos. b. **main()**: La función principal del programa. Muestra un mensaje de bienvenida y un menú de opciones para interactuar con el programa. Procesa la entrada del usuario y llama a las funciones correspondientes según la opción seleccionada.
4. Flujo del programa: El programa sigue el siguiente flujo:
   * Establece una conexión con la base de datos y crea una tabla para almacenar los Pokémon si no existe.
   * Muestra el menú de opciones y solicita al usuario que seleccione una opción.
   * Dependiendo de la opción seleccionada, realiza las siguientes acciones:
     + Opción 1: Solicita al usuario el nombre o número del Pokémon y muestra su información. También da la opción de guardar el Pokémon en un equipo específico.
     + Opción 2: Solicita al usuario el nombre del equipo y muestra los Pokémon guardados en ese equipo.
     + Opción 3: Solicita al usuario el nombre del equipo y el nombre o número del Pokémon a quitar, y elimina el Pokémon del equipo.
     + Opción 4: Sale del programa.
   * Cierra la conexión con la base de datos al salir del programa.
5. Interacción con la base de datos: El programa interactúa con una base de datos llamada "pokedb". Utiliza la función **connect** para establecer la conexión con la base de datos, proporcionando la dirección del servidor, el nombre de usuario, la contraseña y el nombre de la base de datos. Luego, utiliza funciones como **create\_pokemon\_table**, **save\_pokemon\_to\_team** y **remove\_pokemon\_from\_team** para realizar operaciones relacionadas con la tabla de Pokémon en la base de datos.

¿Por qué elegimos la pokeAPI?

1. Datos completos y actualizados: PokéAPI proporciona una amplia gama de datos relacionados con los Pokémon, incluyendo información sobre sus características, movimientos, habilidades, tipos y más. Estos datos están basados en la Pokédex oficial y se actualizan regularmente para mantenerse al día con las nuevas generaciones de Pokémon.
2. Fácil acceso y documentación clara: La API de PokéAPI ofrece una interfaz de programación sencilla y bien documentada, lo que facilita su integración en proyectos de desarrollo. Proporciona endpoints claros y específicos para acceder a los diferentes tipos de datos de Pokémon, lo que simplifica el proceso de obtención de información precisa y detallada.
3. Gratuita y sin necesidad de autenticación: PokéAPI es una API de acceso abierto y gratuito, lo que significa que no es necesario pagar ninguna tarifa ni registrarse para obtener una clave de API. Esto hace que sea conveniente para proyectos de pequeña escala o para desarrolladores que deseen explorar y experimentar con los datos de Pokémon sin restricciones financieras.
4. Comunidad activa y soporte: PokéAPI cuenta con una comunidad de desarrolladores activa y una presencia en GitHub, lo que facilita obtener soporte, realizar consultas y acceder a ejemplos de código. Además, la comunidad contribuye constantemente con mejoras y actualizaciones a la API, lo que garantiza su mantenimiento y continuidad a largo plazo.

Descripción de la base de datos:

1. El archivo **database.py** contiene funciones que interactúan con una base de datos MySQL para realizar operaciones relacionadas con el almacenamiento y recuperación de datos de los equipos de Pokémon. Estas operaciones incluyen la conexión a la base de datos, la creación de una tabla para los equipos, el guardado de Pokémon en un equipo, la obtención de los Pokémon de un equipo y la eliminación de Pokémon de un equipo.
2. Dependencias: El archivo **database.py** depende de la biblioteca **mysql.connector**, que proporciona la funcionalidad para conectarse a una base de datos MySQL y ejecutar consultas SQL.
3. Funciones: a. **connect(host, user, password, database)**: Esta función se utiliza para establecer una conexión con la base de datos MySQL. Recibe los parámetros de conexión, como el nombre de host, el nombre de usuario, la contraseña y el nombre de la base de datos. Devuelve un objeto de conexión que se utiliza para ejecutar consultas y transacciones en la base de datos. b. **create\_pokemon\_table(connection)**: Esta función crea una tabla llamada "team" en la base de datos si no existe. La tabla tiene tres columnas: "pokemon\_number" (cadena de caracteres de longitud máxima 10) que es la clave primaria, "team\_name" (cadena de caracteres de longitud máxima 50) que indica el nombre del equipo y "pokemon\_name" (cadena de caracteres de longitud máxima 50) que indica el nombre del Pokémon en el equipo. c. **save\_pokemon\_to\_team(connection, team\_name, pokemon\_data)**: Esta función guarda un Pokémon en un equipo específico en la base de datos. Recibe la conexión a la base de datos, el nombre del equipo y los datos del Pokémon en forma de diccionario. Inserta una nueva fila en la tabla "team" con el número del Pokémon, el nombre del equipo y el nombre del Pokémon. d. **get\_team\_pokemon(connection, team\_name)**: Esta función recupera los Pokémon de un equipo específico de la base de datos. Recibe la conexión a la base de datos y el nombre del equipo. Ejecuta una consulta SQL para seleccionar los números y nombres de los Pokémon que pertenecen al equipo especificado y devuelve los resultados en forma de una lista de diccionarios. e. **remove\_pokemon\_from\_team(connection, team\_name, pokemon\_id)**: Esta función elimina un Pokémon de un equipo en la base de datos. Recibe la conexión a la base de datos, el nombre del equipo y el identificador (número o nombre) del Pokémon a eliminar. Ejecuta una consulta SQL para eliminar la fila correspondiente en la tabla "team" que coincide con el nombre del equipo y el identificador del Pokémon.
4. Uso de la base de datos: El archivo **database.py** proporciona las funciones necesarias para interactuar con una base de datos MySQL y realizar operaciones relacionadas con los equipos de Pokémon. Se utiliza una tabla llamada "team" para almacenar los Pokémon de los equipos. Cada fila de la tabla representa un Pokémon en un equipo y contiene su número, el nombre del equipo y el nombre del Pokémon.

Explicación de las pruebas que se realizaron en el proyecto

Básicamente las pruebas que le hicimos al proyecto fueron pruebas unitarias.

Estas pruebas se realizan utilizando el módulo **unittest** de Python.

Aquí vemos como se explican las pruebas realizadas para cada función:

1. **test\_connect**: Esta prueba verifica que la función **connect** se llame con los parámetros correctos para establecer una conexión a la base de datos. Se utiliza el decorador **@mock.patch** para simular el comportamiento de la función **mysql.connector.connect** y se comprueba que se haya llamado una vez con los parámetros esperados.
2. **test\_create\_pokemon\_table**: Esta prueba verifica que la función **create\_pokemon\_table** ejecute correctamente la consulta SQL para crear la tabla "team" en la base de datos. Se utiliza el decorador **@mock.patch** para simular el comportamiento de la función **mysql.connector.connect** y se comprueba que se haya llamado una vez a **execute** en el cursor y a **commit** en la conexión.
3. **test\_save\_pokemon\_to\_team**: Esta prueba verifica que la función **save\_pokemon\_to\_team** guarde correctamente un Pokémon en un equipo en la base de datos. Se utiliza el decorador **@mock.patch** para simular el comportamiento de la función **mysql.connector.connect** y se comprueba que se haya llamado una vez a **execute** en el cursor con los parámetros esperados y a **commit** en la conexión.
4. **test\_get\_team\_pokemon**: Esta prueba verifica que la función **get\_team\_pokemon** obtenga correctamente los Pokémon de un equipo en la base de datos. Se utiliza el decorador **@mock.patch** para simular el comportamiento de la función **mysql.connector.connect** y se configura el objeto **cursor** simulado para que devuelva un resultado esperado al llamar a **fetchall**. Se comprueba que se haya llamado una vez a **execute** en el cursor con los parámetros esperados y a **fetchall** en el cursor. Además, se verifica que el resultado obtenido de la función sea igual al resultado esperado.
5. **test\_remove\_pokemon\_from\_team**: Esta prueba verifica que la función **remove\_pokemon\_from\_team** elimine correctamente un Pokémon de un equipo en la base de datos. Se utiliza el decorador **@mock.patch** para simular el comportamiento de la función **mysql.connector.connect** y se comprueba que se haya llamado una vez a **execute** en el cursor con los parámetros esperados, a **commit** en la conexión y a **close** en el cursor.

Conclusiones:

Después del desarrollo del proyecto y en base a las pruebas unitarias realizadas, se llego a la conclusión que dichas pruebas permiten verificar que cada función en el archivo database.py funcione correctamente. Al ejecutar las pruebas, se puede confirmar si las funciones se comportan como se espera y realizan las operaciones adecuadas en la base de datos.

Y al final concluimos que al utilizar las pruebas unitarias en el proyecto se obtiene una mayor confiabilidad en las funciones del archivo **database.py**, se detectan errores de forma temprana, se facilita el mantenimiento y refactoring del código, y se proporciona documentación adicional sobre el uso y comportamiento de las funciones. Esto contribuye a la calidad y robustez general del proyecto.