Breve Informe:

1. **¿Qué API(s) usaste?**  
   En este código se utilizó la API pública **PokeAPI** (https://pokeapi.co/api/v2/pokemon/{pokemonName}), que proporciona información sobre Pokémon, como peso, altura, tipo y sprites (imágenes).
2. **¿Qué dificultades encontraste y cómo las resolviste?**
   * **Dificultad al manejar datos JSON complejos:** La estructura de la respuesta de la API es anidada, lo que puede ser confuso al acceder a propiedades específicas como types[0]["type"]["name"]. Esto se resolvió utilizando la biblioteca **Newtonsoft.Json** para analizar y navegar fácilmente por el JSON.
   * **Manejo de errores:** Si el nombre del Pokémon no es válido o hay problemas de conexión, se genera una excepción. Esto se manejó envolviendo la solicitud en un bloque try-catch y mostrando un mensaje de error al usuario.
   * **Carga de imágenes desde una URL:** Para cargar la imagen del Pokémon, se utilizó el método pictureBox1.Load(imagenUrl). Asegurarse de que la URL no sea nula fue clave para evitar errores.
3. **¿Qué aprendiste con esta tarea?**
   * Cómo consumir una API REST utilizando HttpClient en C#.
   * Cómo analizar y extraer datos de un JSON utilizando la biblioteca **Newtonsoft.Json**.
   * La importancia de manejar excepciones para mejorar la experiencia del usuario.
   * Cómo trabajar con controles de Windows Forms, como PictureBox y Label, para mostrar datos dinámicos.
4. **¿Cómo te apoyaste de la AI para resolver tus dudas?**  
   La AI fue útil para:
   * Proporcionar ejemplos claros de cómo consumir una API REST con HttpClient.
   * Explicar cómo manejar y navegar por estructuras JSON complejas con **Newtonsoft.Json**.
   * Sugerir buenas prácticas para manejar excepciones y mejorar la robustez del código.
   * Resolver dudas específicas sobre el uso de controles de Windows Forms, como PictureBox