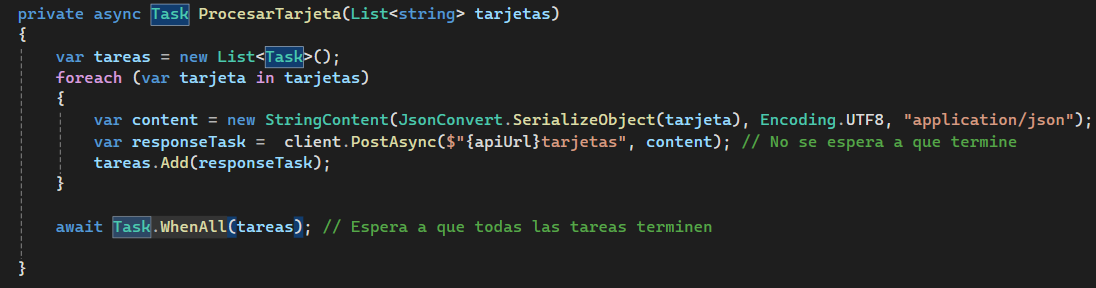
Determinismo: Cuando ejecutamos un método podemos predecir el valor de respuesta. En el caso del paralelismo no siempre es posible determinar el valor de respuesta o el orden de ejecución de los distintos hilos de la aplicación.

**Async – Await**: Me permiten trabajar con hilos sin generar bloqueos, entonces la interfaz de usuario no se bloquea.

**Task**: Es una promesa de que la operación a realizar no necesariamente va a concluir de manera inmediata.

Excepciones: si no se coloca el **await** cuando se hace la llamada a un **Task** la excepción nunca se lanza.

**Await Task.WhenAll**: Espera que terminen todas la tares que se están procesando y luego devuelve los datos. Ejemplo:

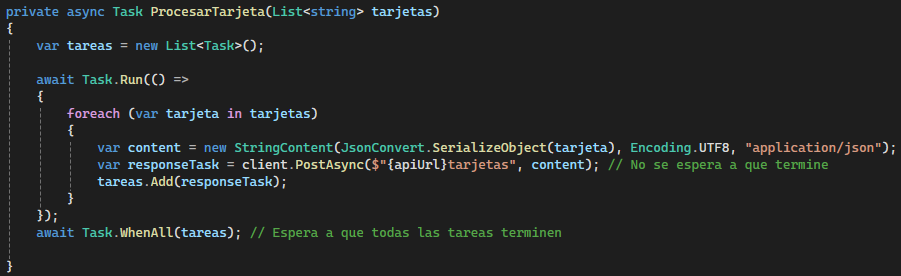


Todo lo que se está procesando dentro del foreach se hace dentro del hilo ‘**UI**’ y hasta que no se ejecute el **Await Task.WhenAll** permanecerá bloqueado, en este caso el form1 permanece bloqueado.

Estamos abrumando el servicio web enviando muchas más peticiones web de la que este puede manejar (25.000 peticiones una por tarjeta).

¿Cómo evito bloquear el hilo ‘**UI**’ si estoy esperando a que se procese una cantidad importante de información?

Para evitar bloquear el hilo ‘UI’ debemos liberarlo ejecutando esas tareas en otro hilo y para esto utilizamos **Task.Run()** con elque vamos a usar para crear nuestra propia tareas:



Con la implementación anterior liberamos el hilo UI, pero el servidor web recibe más peticiones de las que puede manejar y esto lo demos administrar controlando el flujo del sistema “Throttling”**:**

Como se están generando peticiones a una tasa que el servidor web no puede procesar debemos implementar un control:

