GAÑANADA Nº 004

El catch que se traga hasta las ganas de vivir

. . .

CATEGORÍA	INGENIERÍA DE SCHRÖDINGER
GRAVEDAD	■■■■□ IV — EXTREMO
FECHA DEL HALLAZGO	6 DE OCTUBRE DE 2026
ETIQUETAS	#excepciones #manejo-de-errores #observabilidad #python

§ 1

Descripción del hallazgo

El espécimen vive en un proceso que se ejecuta cada noche a las 02:00 para registrar en el ERP los pagos recibidos durante el día:

```python
def sincronizar_pagos():
    for pago in pagos_pendientes():
        try:
            erp.registrar(pago)
            pago.marcar_sincronizado()
        except:
            pass
```

Cuatro líneas de lógica y dos de silencio. El `except:` sin tipo captura absolutamente todo: errores de red, de autenticación, de programación —un `AttributeError` por un campo renombrado— e incluso `KeyboardInterrupt` y `SystemExit`, que en Python no heredan de `Exception` precisamente para que nadie las capture por accidente.

El espécimen las captura a propósito, sin saberlo.

§ 2

Contexto del descubrimiento

En marzo caducó el certificado de cliente con el que el proceso se autenticaba contra el ERP. A partir de esa noche, cada llamada a `erp.registrar` lanzó una excepción; cada excepción fue capturada y descartada; y el proceso terminó con código de salida 0 en unos 400 milisegundos, notablemente más rápido que antes. El panel de monitorización, que solo vigilaba el código de salida, celebró la mejora de rendimiento en verde.

El hallazgo se produjo en octubre, durante el cierre trimestral, cuando el departamento financiero detectó 11 400 pagos cobrados que no figuraban en contabilidad. Un ingeniero ejecutó el proceso a mano para observarlo y descubrió una propiedad adicional del espécimen: no se puede detener con `Ctrl+C`. La interrupción se lanza dentro del `try`, el `except:` se la traga y el bucle pasa amablemente al siguiente pago.

Siete meses. El sistema estaba a la vez funcionando y roto, y nadie abrió la caja hasta que la abrió el cierre trimestral.

§ 3

Consecuencias observadas

- **Fallo invisible.** El tiempo medio de detección fue de siete meses; el de diagnóstico, de diez minutos, una vez que alguien miró.
- **Datos inconsistentes entre sistemas**: cobros reales sin asiento contable, con conciliación manual posterior.
- **Diagnóstico sin evidencia.** Ni trazas, ni mensajes, ni métricas: la información del error se destruyó en el momento en que se produjo.
- **Monitorización engañada.** El proceso cumplía exactamente la definición de éxito que se le había dado.

§ 4

Recomendación de la academia

Captura solo las excepciones que sabes tratar, en el nivel donde puedes hacer algo útil con ellas. Todas las demás deben propagarse: un proceso que falla con estrépito es un problema; uno que falla en silencio es una fuga.

```
def sincronizar_pagos() -> int:
    fallidos = 0
    for pago in pagos_pendientes():
        try:
            erp.registrar(pago)
        except ErpNoDisponible:
            log.warning("ERP no disponible; se reintentará", extra={"pago":
pago.id})
            fallidos += 1
            continue
        pago.marcar_sincronizado()

    metricas.gauge("pagos.pendientes", contar_pendientes())
    return fallidos


if __name__ == "__main__":
    sys.exit(1 if sincronizar_pagos() else 0)
```

En esta versión, un fallo transitorio queda registrado con contexto, el proceso termina con un código que refleja la realidad y cualquier error inesperado —un certificado caducado, un campo renombrado— detiene la ejecución con su traza completa.

Además:

- **Monitoriza resultados, no ausencia de errores.** La métrica útil era «pagos pendientes de más de 24 horas», y habría saltado la primera noche.
- **Prohíbe la especie en el linter**: E722 (*bare except*) en pycodestyle, BLE001 en Ruff, `catch (Exception e) {}` vacíos en Java y `catch (...) {}` en C++ con las reglas equivalentes de cada analizador.

- **Si de verdad hay que capturarlo todo** —en el borde de un proceso, por ejemplo—, registra la traza completa y vuelve a lanzar o termina con error.

No trabajes así, no seas un gañán.

REFERENCIAS

1 Python — tutorial oficial, «Errors and Exceptions»

2 Ruff — regla BLE001 («blind-except») y pycodestyle E722 («bare except»)

Separata del volumen I de la Bitácora del Gañán, publicada por AXXA.
https://axxa.embeddedsource.es/bitacora-del-ganan/004-el-catch-que-se-traga-hasta-las-ganas-de-vivir