

# **STRATÉGIE DE LOGS EN**

## **C#/NET**

### **STOP AUX Error POUR DU FONCTIONNEL**

Ressources & version officielle : [nicolas-cousin.com](http://nicolas-cousin.com)

# SYMPTÔME

- Des règles métier génèrent des Error
- Conséquence : alertes polluées, équipes désensibilisées, enquêtes plus lentes

# RÈGLE D'OR (NIVEAU = ACTION ATTENDUE)

- Information : événement normal
- Warning : anomalie attendue / récupérable
- Error : échec non récupérable ou impact SLO
- Critical : indisponibilité, sécurité, corruption

# FONCTIONNEL VS TECHNIQUE

**Fonctionnel** → souvent **Warning/Information**

- Règle métier non satisfaite, doublon, idempotence
- Technique** → **Error/Critical**
- Exception non gérée, DB down, timeout, corruption

# EXEMPLES – FONCTIONNEL

```
logger.LogWarning("Payment refused: insufficient funds. OrderId={  
    orderId, customerId);  
  
logger.LogInformation("Order already processed (idempotent). Orde
```

# EXEMPLES – TECHNIQUE

```
catch (SqlException ex)
{
    logger.LogError(ex, "Database failure while confirming order.
    throw;
}
```

# LOGS STRUCTURÉS + EventId

- Propriétés obligatoires : OrderId, CustomerId, CorrelationId/TraceId
- EventId pour classer, filtrer, dashboarder
- Pas de concaténation de strings

# EXAMPLE EventId

```
private static readonly EventId PaymentRefused = new(12010, nameof(PaymentRefused));
private static readonly EventId DbFailure = new(50010, nameof(DbFailure));

logger.LogWarning(PaymentRefused,
    "Payment refused. Reason={Reason} OrderId={OrderId}", reason,
    orderId);

logger.LogError(DbFailure, ex,
    "Database failure. OrderId={OrderId}", orderId);
```

# EXCEPTIONS : SIMPLE RÈGLE

- Cas métier gérés **sans** exception
- Exceptions réservées à l'imprévu technique

# MINI-CHECKLIST (ACTIONNABLE EN ÉQUIPE)

1. Table “niveau par scénario” (fonctionnel vs technique)
2. Catalogue EventId + propriétés standardisées
3. Alerter surtout sur Error/Critical (Warnings ciblés au besoin)
4. Revue Top 20 Error → downgrade si fonctionnel

# MESSAGE FINAL

- Warning = incident **fonctionnel** maîtrisé / récupérable
- Error = incident **technique** ou échec non récupérable
- Moins de bruit → meilleures alertes → moins de temps perdu