

# STRATÉGIE DE LOGS EN C#/.NET

STOP AUX *E r r o r* POUR DU  
FONCTIONNEL

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

# SYMPTÔME

- Des règles métier génèrent des Error
- Résultat :
  - Alerting pollué
  - Équipes “désensibilisées”
  - Investigation plus lente (faux positifs)

# RÈGLE D'OR : NIVEAU = ACTION ATTENDUE

- **Information** : événement normal, attendu
- **Warning** : anomalie **attendue / récupérable**, action possible
- **Error** : échec **non récupérable** pour l'opération (ou SLO impact)
- **Critical** : incident majeur (indisponibilité / sécurité / corruption)

# FONCTIONNEL VS TECHNIQUE

## Fonctionnel (souvent Warning/Info)

- validation KO, règle métier non satisfaite
- idempotence / doublon / “déjà traité”

## Technique (souvent Error/Critical)

- exception non gérée, DB down, timeout
- bug, contrat cassé, corruption

# EXEMPLES CONCRETS – FONCTIONNEL $\Rightarrow$ Warning / Information

```
logger.LogWarning("Payment refused: insufficient funds. OrderId={  
    orderId, customerId);
```

```
logger.LogInformation("Order already processed (idempotent). Orde
```

# EXEMPLES CONCRETS – TECHNIQUE $\Rightarrow$ Error

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

# INDISPENSABLE : STRUCTURÉ + EventId

- Toujours des propriétés (OrderId, CustomerId, CorrelationId)
- EventId pour classer, filtrer, dashboarder
- Pas de concaténation de strings

# EXEMPLE – EventId + PROPRIÉTÉS

```
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);
```



# RAPPEL : EXCEPTIONS

- Éviter `LogWarning(ex, . . .)` pour exceptions techniques
- Pattern recommandé :
  - gérer les cas métier **sans** exception
  - réserver l'exception à l'imprévu/technique

# MINI-CHECKLIST D'ÉQUIPE

1. Définir une table “niveau par scénario”
2. Ajouter EventId (catalogue)
3. Standardiser CorrelationId / TraceId
4. Alerter surtout sur Error/Critical (Warnings ciblés si besoin)
5. Revue “Top 20 logs Error” → downgrade si fonctionnel

# MAPPING RAPIDE (ANNEXE)

- **Information** : flux nominal + audit léger
- **Warning** : anomalie attendue, pas d'astreinte par défaut
- **Error** : échec technique, déclenche investigation
- **Critical** : urgence, indisponibilité, sécurité

# MESSAGE FINAL

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