Rapport de Performance (Template)

par Célian Lebacle

1) Contexte & périmètre

- **Description courte du projet / stack :**
  ex : Next.js + Node, Prisma + PostgreSQL (Neon), Redis optionnel…
- **Endpoints / parcours étudiés (top 2–3) :**
  a. _____ (ex. `/api/search`)
  b.
  c.
- **Hypothèses de goulots (avant mesure) :**
  - DB (indexes, N+1, pool) - [ ] CPU/loop - [ ] Cache - [ ] Réseau - [ ] Autre : _____

2) SLOs (objectifs mesurables)

Endpoint	SLI Latence	Seuil p95	SLI Erreurs	Seuil
/api/_____	p95 HTTP	___ ms	5xx rate	< ___ %
/api/_____	p95 HTTP	___ ms	5xx rate	< ___ %

3) Environnement & reproductibilité

- **Version/tag repo** : `perf-baseline` (commit: _____) ; `perf-after` (commit: _____)
- **Données** : seed/dataset = _____
- **Infra locale / staging** : CPU/RAM, nb d'instances, pool DB : _____
- **Variables** (ports, URLs, DSN) : _____
- **Commandes** pour lancer (app/prometheus/grafana/k6) :

```
# Exemple
docker compose up -d
k6 run ./k6/stress.js
```

4) Instrumentation (cochez ce qui est en place)

- Logs Pino (JSON) + `request_id`
- Endpoint `/metrics` Prometheus (prom-client)
  - Histogramme HTTP (labels: method, route, status_code)
  - Event loop lag gauge (optionnel)
  - DB metrics (optionnel)
- Sentry (traces/erreurs) (optionnel)

Captures (insérer) : */metrics extrait, ex. log JSON corrélé*

5) Dashboard Grafana (baseline)

- **Panels présents** : p95 par route, RPS, 5xx rate, (DB p95/pool si dispo)
- **Fenêtre d'observation** : _____
- **Captures** :
  - **Graph p95 (baseline)** : *(image ici)*
  - **Graph RPS** : *(image ici)*
  - **Graph 5xx** : *(image ici)*
  - (Optionnel) **DB p95 / pool** : *(image ici)*

6) Tests k6

- **Scénarios** (cochez) : [ ] smoke [ ] stress [ ] spike [ ] soak [ ] RPS constant
- **Thresholds** (copiez depuis vos scripts) :

```
export const options = {
  thresholds: {
    http_req_duration: ["p(95)<___"],
    http_req_failed: ["rate<___"],
    // autres…
  }
};
```

- **Commandes exécutées** (avec variables d'env) :

```
k6 run -e BASE_URL=_____ ./k6/___.js
```

- **Captures** : *sortie k6*

7) Analyse (baseline)

- **Constats** (croisez k6 + Grafana) :
  ex : p95 ↑ quand RPS ↑, pool DB ~95%, 5xx spikes à 504, route /api/search dominante…
- **Diagnostic principal** : _____
  (cochez) [ ] index manquant [ ] N+1 [ ] pool [ ] CPU/loop [ ] cache [ ] réseau [ ] autre : _____
- **Preuves** (captures EXPLAIN ANALYZE, logs, traces Sentry) : *(images / extraits)*

8) Plan d'action (max 3 leviers, classés par impact/effort)

1. _____ (ex : index (status, updated_at desc))
2. _____ (ex : pagination keyset)
3. _____ (ex : cache Redis 30s + SWR)

Risques / rollback : _____

9) After (mesures après actions)

- **Conditions identiques** (tag `perf-after`, même dataset) : [ ] Oui [ ] Non (si non, expliquer)
- **Résultats k6** :

```
http_req_duration.....: avg=___ ms  med=___ ms  p(95)=___ ms  p(99)=___ ms
http_req_failed.......: ___%
```

- **Dashboard Grafana (captures "après")** : *(images ici)*

Comparaison claire (avant → après)

Metric	Avant	Après	Δ
p95 /api/___	___ ms	___ ms	___ %
5xx rate	___ %	___ %	___ pp
RPS soutenu	___	___	___ %

10) Décision & suites

- **Objectif SLO atteint ?** [ ] Oui [ ] Non
- **Actions suivantes** (si non) : _____
- **Leviers à tester plus tard** : _____

11) Annexes

- **EXPLAIN ANALYZE** / plans
- **Scripts k6**
- **PromQL** des panels
- **Notes diverses**