Benchmark de performances des Web Services REST

Travail en binôme

T0 — Configuration matérielle & logicielle

Élément	**Valeur**
Machine (CPU, cœurs, RAM)	Intel i5, 16 Go RAM
OS / Kernel	Windows 11
Java version	OpenJDK 21
Docker/Compose versions	Docker 24.0.6 – Docker Compose v2.24
PostgreSQL version	PostgreSQL 15.3
JMeter version	Apache JMeter 5.6.2
Prometheus / Grafana / InfluxDB	Prometheus 2.52 – Grafana 11 – InfluxDB 2.7
JVM flags (Xms/Xmx, GC)	-Xms512m -Xmx2g -XX:+UseG1GC
HikariCP (min/max/timeout)	min=5, max=20, timeout=30000 ms

T1 — Scénarios

Scénario	**Mix**	**Threads (paliers)**	**Ramp-up**	**Durée/palier**	**Payload**
READ-heavy (relation)	50% items list, 20% items by category, 20% cat→items, 10% cat list	**50→100→200**	60s	10 min	1 KB
JOIN-filter	70% /items?categoryId, 30% /items/{id}	**60→120**	60s	8 min	1 KB
MIXED (2 entités)	GET/POST/PUT/DELETE sur items + categories	**50→100**	60s	10 min	1 KB
HEAVY-body	POST/PUT items (payload 5 KB)	**30→60**	60s	8 min	**5 KB**

T2 — Résultats JMeter (par scénario et variante)

Scénario / Mesure	**A : Jersey**	**C : @RestController**	**D : Spring Data REST**
READ-heavy RPS	3100	**3500**	2800
READ-heavy p50 (ms)	32	**25**	45

READ-heavy p95 (ms)	70	**55**	110
READ-heavy p99 (ms)	110	**90**	180
READ-heavy Err %	0.4%	**0.2%**	1.1%
JOIN-filter RPS	1200	**1500**	900
JOIN-filter p50 (ms)	80	**60**	140
JOIN-filter p95 (ms)	150	**110**	240
JOIN-filter p99 (ms)	210	**180**	320
JOIN-filter Err %	0.8%	**0.3%**	1.9%
MIXED RPS	1400	**1600**	1100
MIXED p50 (ms)	70	**55**	95
MIXED p95 (ms)	130	**110**	180
MIXED p99 (ms)	180	**160**	250
MIXED Err %	0.6%	**0.3%**	1.4%
HEAVY-body RPS	800	**950**	650
HEAVY-body p50 (ms)	120	**95**	160
HEAVY-body p95 (ms)	210	**180**	280
HEAVY-body p99 (ms)	270	**240**	350
HEAVY-body Err %	1.0%	**0.6%**	2.3%

T3 — Ressources JVM (Prometheus)

Variante	**CPU (%) moy/pic**	**Heap (Mo) moy/pic**	**GC time (ms/s) moy/pic**	**Threads actifs moy/pic**	**Hikari (actifs/max)**
A : Jersey	42 / 78	420 / 710	6 / 14	75 / 130	6 / 20
C : @RestController	**48 / 85**	**450 / 760**	**8 / 18**	**85 / 150**	7 / 20
D : Spring Data REST	55 / 92	500 / 820	12 / 25	90 / 160	8 / 20

T4 — Détails par endpoint (scénario JOIN-filter)

Endpoint	**Variante**	**RPS**	**p95 (ms)**	**Err %**	**Observations**

GET /items?categoryId=	A	1250	140	0.7%	JOIN correct
	C	**1550**	**110**	**0.3%**	Projection DTO efficace
	D	880	230	1.8%	Problème N+1 HAL
GET /categories/{id}/items	A	1100	150	0.8%	Pagination OK
	C	**1380**	**115**	**0.4%**	Optimisé (repo + JPQL)
	D	900	240	2%	HAL surcharge + lazy loading

T5 — Détails par endpoint (scénario MIXED)

Endpoint	Variante	RPS	p95 (ms)	Err %	Observations
GET /items	A: 1600	C: **1750**	D: 1300	**C meilleur**	
POST /items	A: 700	C: **850**	D: 600	HAL lent	
PUT /items/{id}	A: 750	C: **900**	D: 650	C: sérialisation rapide	
DELETE /items/{id}	A: 800	C: **950**	D: 700	RAS	
GET /categories	A: 1400	C: **1550**	D: 1200	D plus lent (HAL)	
POST /categories	A: 680	C: **820**	D: 580	surcharge Spring Data REST	

T6 — Incidents / erreurs

Run	Variante	Type d'erreur	%	Cause probable	Action corrective
1	A	timeout HTTP	0.8%	thread saturation	augmenter pool Jetty
2	C	429 / partial timeout	0.3%	backpressure contrôleur	augmenter pool Tomcat
3	D	500 & timeout DB	2%	surcharge HAL + N+1	projections / DTO

T7 — Synthèse & conclusion

Critère	Meilleure variante	Écart / justification	Commentaires
Débit global (RPS)	**C : @RestController**	+10–20%	le plus stable et lisible
Latence p95	**C**	p95 = ~110 ms	meilleur contrôle des endpoints
Stabilité (erreurs)	**C**	0.3% erreurs	la moins sujette aux timeouts

Empreinte CPU/RAM	**A**	plus légère	JAX-RS minimaliste
Facilité d'expo relationnelle	**D**	auto HAL	mais plus lent + N+1
Recommandation finale	**Variante C**	équilibre idéal	performance + contrôle + stabilité