Benchmark de performances des Web Services REST

Réalisé par : Dbibih Othmane et Hchad Anass

L'étude a pour but d'évaluer et de comparer trois approches différentes pour le développement de services REST : JAX-RS (Jersey), Spring Boot avec @RestController, et Spring Data REST. L'analyse porte sur plusieurs critères de performance : la latence des requêtes, le débit (nombre de requêtes traitées par seconde), l'utilisation des ressources système, ainsi que le niveau d'abstraction offert par chaque framework.
Tous les tests sont réalisés sur une base de données PostgreSQL identique et exploitent le même modèle métier afin d'assurer une comparaison juste et cohérente.

T0 — Configuration matérielle & logicielle

Élément	Valeur
Machine (CPU, cœurs, RAM)	MSI 13th Gen Intel(R) Core(TM) i5-13420H (4 cœurs / 8 threads, 2.1–2.9 GHz), 16 Go RAM
OS / Kernel	Windows 11 Home 64-bit (10.0 Build 26200.7171)
Java version	OpenJDK 17
Docker/Compose versions	Docker 24.0 + Docker Compose v2.27
PostgreSQL version	15 (image alpine)
JMeter version	5.6.3
Prometheus / Grafana / InfluxDB	Prometheus 2.54 / Grafana 11.3 / InfluxDB 2.7
JVM flags (Xms/Xmx, GC)	-Xms512m -Xmx2048m -XX:+UseG1GC
HikariCP (min/max/timeout)	minIdle=10, maxPoolSize=20, connectionTimeout=30000

T1 — Scénarios de charge (JMeter)

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

T2 — Résultats JMeter (par scénario et variante)

Scénario	Mesure	A : Jersey	C : @RestController	D : Spring Data REST
READ-heavy	RPS	1280	1400	1150
	p50 (ms)	47	40	55
	p95 (ms)	98	82	120
	p99 (ms)	165	145	215
	Err %	0.5%	0.4%	0.8%
JOIN-filter	RPS	1340	1460	1200
	p50 (ms)	42	36	52
	p95 (ms)	88	72	108
	p99 (ms)	155	125	205
	Err %	0.4%	0.3%	0.6%
MIXED (2 entités)	RPS	830	940	730
	p50 (ms)	77	67	97
	p95 (ms)	142	122	182
	p99 (ms)	225	195	285
	Err %	0.7%	0.5%	1.1%
HEAVY-body	RPS	550	620	470
	p50 (ms)	112	97	127
	p95 (ms)	215	185	255
	p99 (ms)	305	275	345
	Err %	0.9%	0.6%	1.3%

T3 — Ressources JVM (Prometheus)

Variante	CPU (% moy/pic)	Heap (Mo)	GC time (ms/s)	Threads actifs	Hikari actifs/max
A : Jersey	30 / 57	740 / 1000	8 / 23	74 / 97	13 / 20
C : @RestController	33 / 62	780 / 1040	7 / 20	80 / 105	15 / 20
D : Spring Data REST	36 / 65	820 / 1140	10 / 27	87 / 112	16 / 20

T4 — Détails par endpoint (JOIN-filter)

Endpoint	Var.	RPS	p95 (ms)	Err %	Observations
GET /items?categoryId=	A	1340	88	0.4	JOIN FETCH ok, faible N+1
	C	1460	72	0.3	Projection DTO rapide
	D	1200	110	0.6	HAL + lazy relations → overhead
GET /categories/id/items	A	1280	92	0.5	Bonne pagination
	C	1410	77	0.3	Caching JSON efficace
	D	1150	112	0.7	Hypermedia ralentit

T5 — Détails par endpoint (MIXED)

Endpoint	Var.	RPS	p95 (ms)	Err %	Observations
GET /items	A	890	138	0.6	Basé sur pagination JPA
	C	970	122	0.5	Moins de sérialisation
	D	760	178	1.1	HAL impacte latence
POST /items	A	800	153	0.7	Validation Bean ok
	C	880	137	0.6	Contrôle fin du mapping
	D	680	192	1.4	Surcoût HAL + DTO générés
PUT /items/id	A	790	157	0.7	Update stable
	C	870	138	0.6	Mise à jour rapide
	D	670	187	1.2	Surcharge Jackson
DELETE /items/id	A	860	142	0.5	Bon rollback
	C	950	122	0.4	API claire
	D	720	172	1.0	Proxy repository
GET /categories	A	880	132	0.5	Pagination simple
	C	950	118	0.4	Bon pool SQL
	D	740	168	1.1	HAL verbose
POST /categories	A	830	147	0.6	Insert simple
	C	910	132	0.5	Bon traitement JSON
	D	720	178	1.0	Validation automatique

T6 — Incidents / erreurs

Run	Variante	Type d'erreur	%	Cause probable	Action corrective
2	D (Spring Data	HTTP 500	1.2	Timeout transaction PUT	Augmenter

Run	Variante REST)	Type d'erreur	%	Cause probable massif	Action corrective hibernate.jdbc.timeout
3	A (Jersey)	HTTP 429	0.4	Trop de threads JMeter → latence DB	Limiter à 150 threads
4	C (Spring MVC)	DB timeout	0.3	Pool épuisé pendant burst	maxPoolSize=25