Grille d’auto-évaluation

Partie 2 : Labs 3, 4 et 5 — LOG430

Niveaux d’évaluation

Chaque critère est évalué selon l’un des quatre niveaux suivants :

— Excellent : Le travail dépasse les attentes, complet, structuré, pertinent et justifié de façon approfondie.

— Suffisant : Le travail répond aux attentes de base avec des justifications claires.

— Suffisant avec améliorations : Le travail montre des efforts mais présente des lacunes ou manque de clarté.

— Insuffisant : Le critère est absent, incorrect ou mal exécuté.

Grille d’évaluation détaillée

Critère	Excellent	Suffisant	Suffisant avec améliorations	Insuffisant
1. Extension API RESTful (Labo 3)	API complète et stateless, respect du pattern MVC ou hexagonal, routes cohérentes, couche métier isolée, tests fonctionnels.	API exposée avec routes claires, séparation de couches à affiner.	Exposition partielle, logique métier mélangée.	Pas d’API REST ou principes non respectés.
2. Documentation Swagger/OpenAPI (Labo 3)	Spécification OpenAPI 3.0 exhaustive, UI intégrée (Swagger UI/Redoc), exemples d’appels, versionnage clair.	Documentation présente, UI minimale, exemples ou versionnage partiels.	Fichier Swagger incomplet, absence d’exemples ou de codes d’erreur.	Aucune documentation ou non conforme à OpenAPI.
3. Sécurité et CORS (Labo 3)	CORS pour toutes origines/méthodes, authentification (Basic/JWT), endpoints protégés.	CORS et auth basiques configurés.	CORS partiel, sécurité superficielle.	Pas de CORS ou endpoints non protégés.
4. Tests & CI/CD (Labo 3)	Tests automatisés (MockMVC/JUnit), collection Postman, CI	Tests unitaires et manuels basiques, CI déclenchée, couverture partielle.	Tests partiels, pipeline instable.	Pas de tests ou CI non fonctionnel.

	continue verte à chaque commit.			
5. Test de charge & observabilité initiale (Labo 4)	Scénarios réalistes, métriques Prometheus, dashboard Grafana complet (4 Golden Signals).	Tests de charge et métriques basiques, dashboard partiel.	Test isolé, métriques non intégrées.	Pas de test de charge ou observabilité.

Critère	Excellent	Suffisant	Suffisant avec améliorations	Insuffisant
6. Load balancing & résilience (Labo 4)	Load balancer (NGINX/HAProxy/Traefik), tests RPS et pannes simulées, tolérance validée.	Balancing configuré, tests basiques.	Balancer présent, tests incomplets.	Pas de load balancing ou configuration incorrecte.
7. Caching des end-points (Labo 4)	Cache (Spring @Cacheable/Redis), règles d'invalidation, tests avant/après, gains documentés.	Cache implémenté, mesures de latence partielles.	Cache ajouté sans tests systématiques.	Pas de cache ou mauvaise configuration.

8. Découpage microservices (Labo 5)	≥ 4 services conteneurisés, responsabilités claires et uniques (SRP), communication uniquement via API.	4 services conteneurisés, responsabilités globales identifiées.	2–3 services, découpage partiel.	Monolithe, absence de découpage.
9. Organisation DDD	Modèles Domain, Application, Infrastructure clairement définis, Bounded Context identifiés.	DDD esquissé, Contexts identifiés sans détails.	Tentative de DDD, Contexts flous.	Pas d'organisation DDD.
10. Configuration de l'API Gateway	Gateway configurée (routes dynamiques, CORS, auth, logging).	Gateway basique, routes OK.	Gateway statique, middleware manquant.	Pas de Gateway ou configuration non fonctionnelle.
11. Load balancing via Gateway	Réplication des services, balancing opérationnel, comparatif tests sans/avec.	Balancing via Gateway, tests sans analyse détaillée.	Balancing présent, tests limités.	Pas de balancing ou tests non faits.

Critère	Excellent	Suffisant	Suffisant avec améliorations	Insuffisant
12. Logging	Logging centralisé mi(ELK/Promtail/Loki), baniveaux configurés, distribuées.	Logging local ou nimal, niveaux siques. traces	Logging partiel, pas de corrélation.	Pas de logging ou logs inutilisables.

13. Observabilité et comparatif	Dashboards Grafana à jour, comparatif d'architectures avec métriques et captures d'écran.	Dashboards et comparatif basiques.	Observabilité partielle, comparatifs limités.	Aucune observabilité ou comparatif.
14. Rapport d'analyse	Rapport détaillé des optimisations (load balancing, caching), résultats chiffrés, graphiques et recommandations.	Rapport succinct, quelques résultats et graphiques.	Rapport partiel, manque d'analyse critique.	Pas de rapport ou analyse.
15. Documentation et API	Rapport Arc42 structuré, ADR architecturaux inclus, documentation technique à jour, collection Postman et guide de déploiement.	Rapport Arc42 et ADR de base, docs et tests/API partiels.	Documentation superficielle, ADR incomplets.	Aucune documentation, ADR ou tests.

Autoévaluation à compléter par l'étudiant(e)

Nom et prénom : Pravdin Maksym Date : 24 juin 2025

Pour chaque critère, cochez la case correspondant à votre niveau et ajoutez un court commentaire :

Critère 1. Extension API RESTful (Labo 3)

□ Excellent ✓ Suffisant □ Suffisant avec améliorations □ Insuffisant

Commentaire : L'API a été modifié, mais va être amélioré encore plus

Dans le Labo 5 une fois que tous les services sont mis en place pour

Faire une séparation plus approfondie et focusée sur le DDD.

Critère 2. Documentation Swagger/OpenAPI (Labo 3)

✓ Excellent □ Suffisant □ Suffisant avec améliorations □ Insuffisant

Commentaire : Swagger est configuré pour toutes les routes, contient

Les données nécessaires pour chaque route pour les tester et les

informations pour chaque route sont incluses aussi

Critère 3. Sécurité et CORS (Labo 3)

✓ Excellent □ Suffisant □ Suffisant avec améliorations □ Insuffisant

Commentaire : Toutes les routes sont protégées avec la validation du

token qui est stoqué sur le serveur de mise en cache Redis qui valide

si la requête a les accès nécessaires pour faire l'opération, de plus un

CORS est mis en place qui accepte les appels que de

Postman/Swagger et du serveur Django frontend

Critère 4. Tests & CI/CD (Labo 3)

□ Excellent □ Suffisant ✓ Suffisant avec améliorations □ Insuffisant

Commentaire : Il manque encore des tests pour certaines routes

Critère 5. Test de charge & observabilité initiale (Labo 4)

✓ Excellent □ Suffisant □ Suffisant avec améliorations □ Insuffisant

Commentaire : Prometheus+Grafana recoltent et affichent les

métriques de toutes les instances de serveur Express dans un

dashboard personnalisé et toutes les métriques de 4 Golden Signals

sont inclus et k6 est utilisée pour les tests

Critère 6. Load balancing & résilience (Labo 4)

□ Excellent ✓Suffisant □ Suffisant avec améliorations □ Insuffisant

Commentaire : Load balancing est fait avec NGINX et les tests de charge avec k6 sont faits également

Critère 7. Caching des endpoints (Labo 4)

□ Excellent ✓Suffisant □ Suffisant avec améliorations □ Insuffisant

Commentaire : Caching le plus important est fait comme les accès aux tokens car ils sont nécessaires pour presque chaque opération et c'est fait avec un serveur Redis. Redis est mocke dans les tests.

Critère 8. Découpage microservices (Labo 5)

□ Excellent ✓ Suffisant □Suffisant avec améliorations □ Insuffisant

Commentaire : Le système possède 4 types de services différents : Auth, stocks (2 instances + load balancer), sales et supplies.

Critère 9. Organisation DDD (Labo 5)

□ Excellent ✓Suffisant □ Suffisant avec améliorations □ Insuffisant

Commentaire : DDD documenté et amélioré depuis l'étape 1

Critère 10. Configuration de l'API Gateway (Labo 5)

✓Excellent □ Suffisant □ Suffisant avec améliorations □ Insuffisant

Commentaire : Fait avec krakend et est configuré pour toutes les routes et accepte uniquement les appels avec CORS, nécessite les bons headers pour faire la redirection au service.

Critère 11. Load balancing via Gateway (Labo 5)

□ Excellent ✓ Suffisant □ Suffisant avec améliorations □Insuffisant

Commentaire : Load balancing fait avec NGINX, car krakend ne Supporte pas load balancing avec la configuration « no-op » ce qui Necessite ce workaround pour assurer qu'il y a du load balacing pour

les instances de services stocks.

Critère 12. Logging (Labo 5)

□ Excellent ✓Suffisant □ Suffisant avec améliorations □ Insuffisant

Commentaire : Logging fait avec le package ‘pino’ de npm à travers l’ensemble du système.

Critère 13. Observabilité et comparatif (Labo 5)

✓ Excellent □ Suffisant □Suffisant avec améliorations □ Insuffisant

Commentaire : Comparaison faite entre le labo 4, labo 5 avec appels Directs et labo 5 avec gateway/load balancer dans le rapport sous forme de paragraphe, de graphiques Grafana et de tableau.

Critère 14. Rapport d’analyse (Labo 5)

✓ Excellent □ Suffisant □ Suffisant avec améliorations □ Insuffisant

Commentaire : Rapport suit le format ARC42 et contient un historique de toutes les versions du laboratoire ainsi que les explications pertinentes et remarques pour chaque laboratoire

Critère 15. Documentation et API (Labo 5)

✓Excellent □ Suffisant □ Suffisant avec améliorations □ Insuffisant

Commentaire : Swagger configuré pour toutes les routes et avec la Documentation et lancement nécessaire.