Grille d’Autoévaluation - Labo 2 LOG430

# Niveaux d’évaluation

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

* **Excellent** : Le travail est dépasses aux attents, complet, structuré, pertinent et justifié de façon approfondie.
* **Suffisant** : Le travail répond aux attents de base avec des justifications claires.
* **Suffisant avec améliorations** : Le travail montre des efforts mais présente des la- cunes 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. Analyse de l’architecture existante (Labs 0 et 1)** | Analyse critique complète, bien structurée, avec commentaires pertinents | Résumé clair des éléments antérieurs, avec une bonne compréhension | Résumé partiel, sans justification claire ou analyse superficielle | Absence de synthèse ou résumé imprécis et sans analyse |
| **2. Compréhension des exigences fournies (MoSCoW)** | Compréhension claire et complète des exigences fournies | Compréhension généralement correcte, avec quelques imprécisions ou confusions mineures | Compréhension partielle ou approximative, avec confusions | Exigences mal comprises ou non prises en compte dans l’analyse |
| **3. Proposition d’architecture (conception globale)** | Architecture pertinente, cohérente, justifiée, tenant compte des con- traintes | Architecture raisonnable et justifiée de manière adéquate | Architecture partiellement adaptée ou trop complexe sans justification | Architecture incohérente, non adaptée aux exigences. |
| **4. Diagrammes UML (modèle 4+1)** | Diagrammes complets, lisibles, cohérents entre eux, avec légendes | Diagrammes UML présents, corrects, et décrivant bien le système | Diagrammes incomplets, désorganisés ou difficiles à lire | Diagrammes absents ou non pertinents. |
| **5. ADRs (Architectural Decision Records)** | Deux ADRs ou plus, bien rédigés, structurés et justifiés. | Deux ADRs présents, bien structurés mais manquant de justification ou détails | Un seul ADR ou plusieurs superficiels, justification limitée | Aucun ADR ou documents incompréhensibles |
| **6.** **Application des principes du Domain-Driven Design (DDD)** | Les sous-domaines sont bien définis (ex. ventes, logistique), les choix DDD sont clairs (Bounded Contexts, Entities, etc.) et bien intégrés à l’architecture. Des patrons DDD (Aggregates, Entities, Repositories, etc.) sont appliqués de manière cohérente. | Les sous-domaines sont identifiés et modélisés avec justesse. Les choix sont globalement pertinents, même si certains aspects DDD restent superficiels ou non explicités. | Une tentative d’identification des sous-domaines est présente, mais elle manque de clarté ou de justifications. Les concepts DDD sont peu ou mal appliqués. | Aucune trace d’une démarche DDD, ou les concepts sont mal compris ou inadaptés au contexte. |
| **7. Implémentation du prototype** | Prototype complet, stable, valide les choix architecturaux et couvre les fonctionnalités Must et Should. Tous les niveaux de testes automatisés sont ajoutés. Pattrons de projet bien utilisés | Prototype fonctionnel, couvrant les fonctionnalités Must, validant globalement les choix. Testes unitaires et intégrations simples. Pattrons parciallement utiliser | Prototype partiellement fonctionnel, avec instabilités ou lacunes dans les fonctionnalités essentielles. Structure ad-hoc | Implémentation absente, non fonctionnelle ou ne couvrant pas les exigences essentielles. Sans structures ou pattrons des projets appliqués |
| **8. Intégration CI/CD et conteneurisation** | Pipeline automatisé, bien documenté, utilisé activement dans le projet. Containers bien utilisé avec automation des images et deploiement (ex. : Docker Compose) | Pipeline CI/CD en place, fonctionnel et utilisé. Imagens existentes et automatisés | Pipeline présent mais incomplet ou non maintenu | Aucune trace d’automatisation ou CI/CD inexploitable |
| **9. Rapport** | Rapport claire, détaillée, bien struc-turée, expliquant tout le projet | Rapport informatif, couvrant les points essentiels avec instructions de déploiement | Rapport minimale, man-quant d’organisation ou d’explications | Rapport absent, désorganisée ou inutilisable |

# Autoévaluation à compléter par l’étudiant(e)

**Nom et prénom :** Maksym Pravdin **Date :** 06/06/2025

Veuillez compléter cette section en cochant votre niveau estimé pour chaque critère, et en ajoutant un court commentaire justificatif.

## Critère 1 : Analyse de l’architecture existante (Labs 0 et 1)

Excellent Suffisant Suffisant avec améliorations Insuffisant

*Commentaire : Le rapport est écrit en suivant le modèle ARC42 et contient un historique de versions pour garder les analyses des versions précédentes et de la version courante.*

## Critère 2 : Compréhension des exigences fournies (MoSCoW)

Excellent Suffisant Suffisant avec améliorations Insuffisant

*Commentaire : Tous les cas d’utilisation essentiels sont implémentés, plus que la moitié des cas d’utilisations souhaitables sont implémentés (il manque que la revue des demandes d’approvisionnement à partir de la base de données), mais aucun cas d’utilisation facultatif est implémenté, mais le système va avoir une interface Web/mobile dans la future version avec Django.*

## Critère 3 : Proposition d’architecture (conception globale)

Excellent Suffisant Suffisant avec améliorations Insuffisant

*Commentaire : L’architecture proposée répond aux exigences architecturalles de l’application et est justifiée dans le rapport.*

## Critère 4 : Diagrammes UML (modèle 4+1)

Excellent Suffisant Suffisant avec améliorations Insuffisant

*Commentaire : Tous les diagrammes UML du modèle 4+1 sont inclus dans le dossier docs, sont présents dans le rapport ainsi que l’explication de chaque diagramme.*

## Critère 5 : ADRs (Architectural Decision Records)

Excellent Suffisant Suffisant avec améliorations Insuffisant

*Commentaire : 2 nouveaux ADRs sont introduit avec cette version du laboratoire dans le dossier docs et dans le rapport ainsi que les anciens ADR.*

## Critère 6 : Application des principes du Domain-Driven Design (DDD)

Excellent Suffisant Suffisant avec améliorations Insuffisant

*Commentaire : Le principe DDD est appliqué dans l’architecture de l’application et le patron MVC est utilisée pour le backend du système.*

## Critère 7 : Implémentation du prototype

Excellent Suffisant Suffisant avec améliorations Insuffisant

*Commentaire : L’application n’a pas d’interface Web/mobile cncore et ne contient pas les tests unitaires pour Express.js (mais contient les tests pour pytest), cela est dû à la récommendation du chargé de laboratoire d’utiliser Express.js au lieu de FastAPI ce qui m’a forcé de migrer une grande partie de mon code Python en JavaScript.*

## Critère 8: Intégration CI/CD et conteneurisation

Excellent Suffisant Suffisant avec améliorations Insuffisant

*Commentaire : Le pipeline CI/CD est composé de 4 étapes qui doivent être faits de manière séquentielle (Linting, tests, image Docker, upload sur Docker Hub. Celui-ci est lancé après chaque push sur la branche principale.*

## Critère 9 : Rapport final

Excellent Suffisant Suffisant avec améliorations Insuffisant

*Commentaire : Le rapport contient toutes les sections du rapport ARC42 :* <https://arc42.org/overview#solution-strategy> *et chaque section est décrite et contient un historique des versions précédentes pour voir l’évolution du système et de l’architecture.*