Labo 2 – Évolution d'une architecture logicielle plus scalable et flexible

Cours : **Architecture Logicielle (LOG430)**

Session : **Été 2025**

Date du laboratoire : **Semaine du 26 mai 2025**

Le génie pour l'industrie

Contexte

Votre système initial (Lab 1), que vous aviez développé et exécuté sur votre machine virtuelle, doit désormais évoluer pour répondre aux besoins d'une entreprise possédant cinq magasins répartis dans des différents quartiers, un centre de logistique et une maison mère pour les fonctions administratives.

Les nouvelles exigences incluent :

— Permettre la gestion simultanée et cohérente de plusieurs magasins.

— Offrir une consultation centralisée des stocks disponibles et des transactions réalisées des les magasins.

— Assurer la synchronisation fiable et cohérente des données entre les différents magasins et la maison mère.

— Rapports consolidés générés pour l'administration depuis la maison mère

— Évolutivité vers une potentielle interface web ou mobile

Dans ce contexte, vous proposerez une nouvelle architecture logicielle mieux adaptée à ces contraintes et vous en développerez un prototype fonctionnel permettant de valider votre solution.

Objectifs d'apprentissage

— Identifier précisément les limites d'une architecture 2-tiers existante

— Concevoir une architecture évolutive pour plusieurs magasins situés dans la même ville, avec un système hébergé en utilisant la VM.

— Documenter et justifier vos décisions à travers des diagrammes UML (modèle 4+1) et des ADR

— Appliquer concrètement des patrons de conception (MVC, Hexagonal, GoF, etc)

— Implémenter un prototype réaliste et fonctionnel

— Avoir des tests automatisés (unitaries, intégration et end-to-end).

— Maintenir et enrichir vos pratiques CI/CD et conteneurisation

Tâches à réaliser

1. Analyse et continuité

— Résumez clairement les solutions développées aux Labs 0 et 1

— Identifiez explicitement les éléments à conserver, modifier ou refactorer

— Présentez clairement les nouvelles exigences et les défis architecturaux.

— Introduisez une réflexion basée sur les principes du **Domain-Driven Design (DDD)** : identifiez les sous-domaines fonctionnels (Ventes en magasin, gestion logistique, supervision par la maison mère).

2. Proposition d'architecture (libre choix avec justification)

- Concevez une architecture en visant la solution la plus simple possible répondant aux besoins exprimés.
- Justifiez vos décisions à travers au moins deux ADR (Architectural Decision Records)
- Produisez les diagrammes UML (4+1) suivants :
  - **Vue logique**.
  - **Vue processus**.
  - **Vue implémentation**.
  - **Vue déploiement**.
  - **Vue cas d'utilisation**.

3. Exigences fonctionnelles exprimées selon MoSCoW

- **Must have (Essentiel)** :
  - **UC1 – Générer un rapport consolidé des ventes** : Un gestionnaire à la maison mère génère un rapport détaillé contenant les ventes par magasin, les produits les plus vendus, et les stocks restants. Ce rapport est utilisé pour la planification et les décisions stratégiques.
  - **UC2 – Consulter le stock central et déclencher un réapprovisionnement** : Un employé d'un magasin consulte le stock disponible dans le centre logistique. Si un produit est insuffisant localement, il peut initier une demande d'approvisionnement depuis son interface.
  - **UC3 – Visualiser les performances des magasins dans un tableau de bord** : Un gestionnaire de la maison mère accède à un tableau de bord synthétique affichant les indicateurs clés : chiffre d'affaires par magasin, alertes de rupture de stock, produits en surstock, tendances hebdomadaires.
- **Should have (Souhaitable)** :
  - **UC4 – Mettre à jour les produits depuis la maison mère** : Un responsable modifie les informations d'un produit (nom, prix, description). Les changements sont synchronisés automatiquement dans tous les magasins afin d'assurer une cohérence dans les points de vente.
  - **UC6 – Approvisionner un magasin depuis le centre logistique** : Le responsable logistique valide une commande de réapprovisionnement pour un magasin donné. L'opération déclenche le transfert du stock et met à jour les niveaux de stock dans les deux entités.
- **Could have (Facultatif)** :
  - **UC7 – Alerter automatiquement la maison mère en cas de rupture critique** : Lorsqu'un produit atteint un seuil critique de stock dans un ou plusieurs magasins, une alerte automatique est envoyée à la maison mère afin de permettre une action rapide (commande urgente, redistribution).

— **UC8 – Offrir une interface web minimale pour les gestionnaires** : Une interface web légère permet aux gestionnaires d'accéder à distance aux indicateurs clés du système : ventes, stocks, alertes. Elle offre une visibilité rapide sans devoir accéder directement au système interne.

Livrables attendus (organisation du dépôt)

Votre dépôt Git final doit clairement refléter la continuité entre les Labs 0, 1 et 2. Chaque laboratoire doit être documenté et organisé dans son propre dossier. Pour le travail final, vous devez produire un rapport structuré suivant le format Arc42. Ce rapport présentera votre propre proposition d'architecture en visant la solution la plus simple possible répondant aux besoins exprimés. Il devra inclure la justification de vos décisions à travers au moins deux Architectural Decision Records (ADR), ainsi que les diagrammes UML du modèle 4+1 (cas d'utilisation, logique, développement, déploiement, etc.).

Enfin, vous devez joindre un fichier .zip contenant l'ensemble du code source des Labs 0, 1 et 2, organisé proprement et prêt à être exécuté.

Les éléments à fournir sont les suivants :

— **Lab 0 :** Base initiale du projet, comprenant le code source initial, la configuration Docker et Docker Compose, ainsi qu'une documentation initiale claire.

— **Lab 1 :** Application 2-tiers complète avec persistance, tests automatisés, pipeline CI/CD opérationnelle, documentation technique avec diagrammes UML et ADR.

— **Lab 2 :** Architecture évoluée (plusieurs magasins, maison mère). Vous devez fournir :
  — Nouveau code source structuré et fonctionnel.
  — Diagrammes UML détaillés selon le modèle 4+1.
  — Au moins deux ADR justifiant vos choix architecturaux.
  — Un rapport expliquant clairement le projet complet, les technologies choisies, la structure du projet et des instructions d'exécution détaillées.
  — Pipeline CI/CD complète et documentée.

— **Un fichier** `.zip` contenant l'ensemble du code source des Labs 0, 1 et 2, organisé proprement.

Vous pouvez inclure dans votre livrable final (lab 2) les liens vers vos dépôts GitLab ou GitHub distincts, si vous avez opté pour créer des dépôts différents pour chaque laboratoire (Lab 0, Lab 1, Lab 2). Dans ce cas, indiquez clairement chaque lien dans votre rapport du lab 2.

Conseils pédagogiques

— Avancez par étapes progressives, en intégrant clairement les livrables précédents (utilisez des tags Git pour chaque laboratoire)

— Privilégiez une solution réaliste, cohérente et clairement justifiée plutôt qu'une solution trop complexe

— Documentez systématiquement vos choix techniques et architecturaux à chaque étape
— Testez régulièrement votre solution, et intégrez vos tests dans la pipeline CI/CD dès le début