Cahier des charges

Projet : LegoFactory – Outil de gestion d'emplacement de Lego

Auteurs : Etienne Corentin, Léculée-Ravé Mathias, Yousri Bouchrika

Formation : BTS SIO – 2025/2026

1. Contexte

L'entrepôt LegoFactory possède plus de 30 000 sets Lego.

Actuellement :

- Les emplacements physiques sont gérés manuellement dans un cahier.
- La gestion des stocks (quantité, date d'entrée, fournisseurs, etc.) se fait dans un outil distinct, non relié à la gestion des emplacements.
- Cette organisation provoque des problèmes de fiabilité, de recherche et de mise à jour, particulièrement pour les nouveaux arrivants.

2. Objectifs

- Développer une application en C# permettant de numériser et centraliser la gestion des emplacements.
- Fiabiliser et accélérer la recherche des sets.
- Assurer la synchronisation avec l'outil de gestion de stock existant (import/export).
- Optimiser la logistique de l'entrepôt (ajouts, retraits, mouvements).
- Faciliter la traçabilité des sets (localisation, historique).

3. Description fonctionnelle

3.1 Gestion des emplacements

- Création d'une structure hiérarchique : zones → allées → étagères → niveaux.
- Visualisation de l'entrepôt (schéma ou tableau).
- Ajout / modification / suppression d'emplacements.
- Définition d'une capacité maximale par emplacement.
- Alerte en cas de dépassement de capacité.

Génération automatique d'un code d'emplacement unique selon la règle suivante :

- L'**étagère** est représentée par une lettre alphabétique (A, B, C, ...).
- L'**étage** est représenté en centaines (Étage 1 = 100, Étage 2 = 200, etc.).
- La **rangée** est ajoutée en unité (ex. rangée 3 = ...3).

➞ Exemple : *A103 = Étagère A, Étage 1, Rangée 3*.

3.2 Association set ↔ emplacement

- Associer un set Lego à un ou plusieurs emplacements.
- Déplacer un set d'un emplacement à un autre.
- Historique des déplacements.
- Recherche par nom / référence / collection (pour retrouver l'emplacement).
- Recherche par emplacement (pour connaître son contenu).

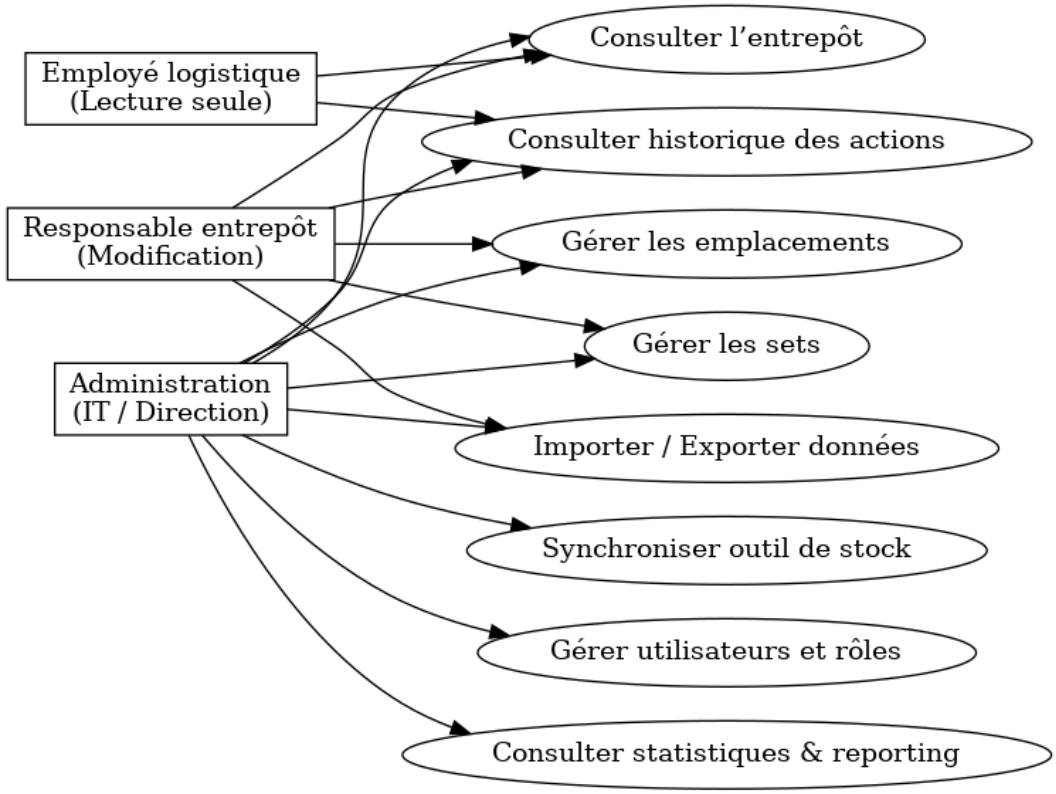
3.3 Import / Export des données

- Import depuis l'outil de stock (CSV / Excel / API).
  - Champs : nom, référence, collection, quantité, code unique, etc.
- Export des positions actuelles (fichier ou API).
  - Données : nom, quantité, emplacement, date d'entrée, etc.

3.4 Synchronisation avec l'outil de stock

- Liaison via une clé unique (code set).
- Mise à jour automatique des emplacements depuis l'outil de stock (API ou import planifié).
- Mise à jour des quantités après ajout/retrait/mouvement.

3.5 Sécurité et droits d'accès

- Authentification des utilisateurs.
- Gestion des rôles :
  - 

- Historique des actions (traçabilité des mouvements).

3.6 Statistiques et reporting

- Nombre de sets par collection / thème / zone.
- Identification des emplacements vides ou sous-utilisés.
- Alertes en cas de déséquilibre entre stock théorique et réel.

4. Contraintes techniques

- Développement sous forme de client lourd.
- Liaison possible avec une base de données externe.
- Compatibilité avec fichiers CSV/Excel et éventuellement une API.
- L'application sera fait en C#

5. Organisation & Ressources

- Équipe : 3 développeurs.
  - **Lead développeur :** Léculée-Ravé – rôle : organisation du planning et attribution des tâches.
    - Taux journalier : 96 € / jour
    - Salaire mensuel estimé : **3 100 €**
  - **Programmeur 1 :** Etienne Corentin – rôle : développement en C#.
    - Taux journalier : 64 € / jour
    - Salaire mensuel estimé : **2 100 €**
  - **Programmeur 2 :** Yousri Bouchrika – rôle : développement en C#.
    - Taux journalier : 64 € / jour
    - Salaire mensuel estimé : **2 100 €**

Total masse salariale mensuelle : **5 300 €**

6. Planning & Délais

- Durée estimée : moins d'un mois.
- Objectif : retour sur investissement (ROI) avant mai.
- ROI prévu : après 9 mois d'utilisation.