Rapport de Projet - Application MatChantier : Gestion des Approvisionnements de Chantier

1. Introduction

Ce rapport de projet présente une analyse détaillée de l'application **MatChantier**, un système de gestion automatique des approvisionnements spécifiquement conçu pour les chantiers de construction. Développée en Java natif, cette application vise à optimiser les processus de gestion de stock pour les magasiniers. Une caractéristique fondamentale du projet est son autonomie, garantissant un fonctionnement sans dépendance à des infrastructures de base de données externes ou à une connexion internet. L'intégration d'une base de données embarquée SQLite assure la persistance des données directement au sein de l'environnement applicatif, facilitant ainsi le déploiement et la portabilité. Ce document formalise les objectifs, les exigences fonctionnelles, la structure architecturale envisagée, ainsi que les directives d'installation et d'exécution, servant de référence principale pour le développement et la mise en œuvre du système.

2. Objectifs du Projet

L'objectif principal de l'application MatChantier est de doter le magasinier d'un outil informatique **intuitif, robuste et efficace** pour la gestion quotidienne des ressources matérielles sur un chantier. Les objectifs spécifiques comprennent :

- **Optimisation de la Gestion des Stocks :** Permettre un suivi rigoureux et complet des articles et matériaux en stock, en minimisant les erreurs de comptabilité et en améliorant la précision des inventaires.
- **Rationalisation des Mouvements de Stock :** Fournir une interface simple pour l'enregistrement rapide et précis des entrées (réceptions) et des sorties (consommations) de matériaux, assurant une traçabilité complète.
- **Prévention des Ruptures et Surstocks :** Mettre en place un système d'alertes basé sur des seuils minimaux de stock pour anticiper les besoins en approvisionnement et éviter les interruptions de travail dues à des ruptures de stock.
- **Fiabilisation des Inventaires Physiques :** Faciliter la réalisation des inventaires en simplifiant la saisie des quantités réelles et en automatisant la comparaison avec le stock théorique, permettant des ajustements précis.
- **Amélioration de la Traçabilité et de l'Analyse :** Générer des rapports détaillés et exportables sur l'état des stocks, les mouvements et les inventaires, offrant ainsi des outils d'analyse pour une meilleure prise de décision.

- **Autonomie Opérationnelle :** Assurer que l'application puisse fonctionner de manière indépendante sur la machine du magasinier, sans nécessiter de serveur de base de données distant ou de connexion Internet, garantissant la continuité des opérations même en environnement de chantier isolé.

3. Exigences Fonctionnelles Détaillées

Les exigences fonctionnelles décrivent les actions et capacités spécifiques que le système doit offrir à l'utilisateur, le magasinier.

3.1. Gestion des Articles / Matériaux (RF1)

Le logiciel doit permettre au magasinier de gérer les informations relatives aux articles et matériaux.

- **RF1.1 : Création d'Articles :** Le magasinier doit pouvoir **ajouter de nouveaux articles/matériaux** en spécifiant les informations suivantes :
  - **Nom de l'article** (obligatoire)
  - Description détaillée
  - **Unité de mesure** (ex: Sac, Tonne, Mètre cube, Litre, Pièce, Rouleau)
  - **Quantité minimale de stock** (seuil d'alerte, obligatoire)
  - Prix unitaire standard (pour référence, non modifiable lors des transactions)
  - **Catégorie** (ex: Gros œuvre, Second œuvre, Électricité, Plomberie, Outils)
  - **Numéro de référence/code unique** (automatiquement généré ou manuel)
- **RF1.2 : Consultation d'Articles :** Le système doit **afficher la liste de tous les articles** disponibles avec leurs détails (nom, description, unité, stock actuel, seuil d'alerte).
- **RF1.3 : Modification d'Articles :** Le magasinier doit pouvoir **modifier les informations** d'un article existant (à l'exception du numéro de référence si celui-ci est généré automatiquement).
- **RF1.4 : Suppression/Désactivation d'Articles :** Le magasinier doit pouvoir **supprimer un article** *uniquement si aucune transaction (entrée ou sortie) ne lui est associée*. Si des transactions existent, l'article doit être **désactivé** pour qu'il n'apparaisse plus dans les sélections futures, mais son historique soit conservé.

3.2. Gestion des Entrées de Stock (Réceptions) (RF2)

Le logiciel doit permettre d'enregistrer les arrivées de matériaux sur le chantier.

- **RF2.1 : Enregistrement des Réceptions :** Le magasinier doit pouvoir **enregistrer l'arrivée de nouveaux articles** avec les informations suivantes :
  - **Article concerné** (sélectionné depuis la liste des articles existants)
  - **Quantité reçue** (obligatoire)

    - **Date et heure de réception** (automatique par défaut, modifiable)
    - Nom du fournisseur (obligatoire)
    - Numéro de bon de livraison (facultatif, pour référence)
    - Commentaires (facultatif)
- **RF2.2 : Mise à jour automatique du Stock :** Le **stock de l'article doit être automatiquement incrémenté** suite à l'enregistrement d'une réception.
- **RF2.3 : Consultation des Historiques de Réception :** Le système doit **afficher la liste de toutes les réceptions** enregistrées, avec la possibilité de filtrer par article, date ou fournisseur.

3.3. Gestion des Sorties de Stock (Consommations) (RF3)

Le logiciel doit permettre d'enregistrer les consommations de matériaux sur le chantier.

- **RF3.1 : Enregistrement des Sorties :** Le magasinier doit pouvoir **enregistrer la sortie d'articles** du stock pour utilisation, avec les informations suivantes :
    - **Article concerné** (sélectionné depuis la liste des articles existants)
    - **Quantité sortie** (obligatoire, la quantité saisie **ne doit pas dépasser le stock disponible**)
    - **Date et heure de sortie** (automatique par défaut, modifiable)
    - **Service / Zone d'affectation** (ex: Fondations, Maçonnerie, Équipe A, Bureau technique, obligatoire)
    - Nom de la personne qui retire l'article (obligatoire)
    - Raison de la sortie (facultatif, ex: besoin urgent, régularisation)
- **RF3.2 : Mise à jour automatique du Stock :** Le **stock de l'article doit être automatiquement décrémenté** suite à l'enregistrement d'une sortie.
- **RF3.3 : Consultation des Historiques de Sortie :** Le système doit **afficher la liste de toutes les sorties** enregistrées, avec la possibilité de filtrer par article, date ou service/zone.

3.4. Consultation et Alertes de Stock (RF4)

Le logiciel doit fournir des outils de visualisation et d'alerte pour le stock.

- **RF4.1 : Vue d'Ensemble du Stock :** Le système doit afficher un **tableau de bord ou une liste claire de tous les articles** avec leur stock actuel.
- **RF4.2 : Alertes de Stock Bas :** Le système doit **signaler visuellement** (ex: couleur rouge, icône d'alerte) les articles dont le stock actuel est inférieur ou égal à leur **seuil d'alerte** (Quantité minimale de stock).
- **RF4.3 : Filtrage des Articles en Stock Bas :** Le magasinier doit pouvoir **filtrer la liste des articles** pour n'afficher que ceux dont le stock est bas.

3.5. Rapports (RF5)

Le logiciel doit permettre la génération de rapports imprimables ou exportables.

- **RF5.1 : Rapport de Stock Actuel :** Le système doit **générer un rapport** (imprimable / exportable en **PDF**, **CSV**) du stock actuel de tous les articles ou d'une sélection d'articles.
- **RF5.2 : Rapport d'Historique des Mouvements :** Le système doit **générer un rapport** (imprimable / exportable en **PDF**, **CSV**) de l'historique des entrées et/ou sorties, avec des options de filtrage par période et/ou par article.
- **RF5.3 : Rapport d'Alertes :** Le système doit **générer un rapport** des articles dont le stock est bas.

3.7. Interface Utilisateur (IHM) (RF7)

L'interface du logiciel doit être conçue pour une utilisation intuitive par le magasinier.

- **RF7.1 : Navigation Intuitive :** L'interface doit être **claire, simple et facile à utiliser**, même pour un utilisateur ayant des compétences informatiques de base.
- **RF7.2 : Validation des Saisies :** Le système doit **valider les données saisies** par le magasinier (ex: format numérique pour les quantités, champs obligatoires remplis) pour prévenir les erreurs.
- **RF7.3 : Confirmation des Actions :** Le système doit **demander une confirmation** avant d'effectuer des actions irréversibles (ex: suppression d'article si aucune transaction).
- **RF7.4 : Messages d'Erreur et de Succès :** Le système doit afficher des **messages clairs** informant le magasinier du succès ou de l'échec des opérations.

3.8. Gestion des Inventaires (RF8)

Le logiciel doit permettre au magasinier de réaliser et de suivre les inventaires physiques.

- **RF8.1 : Création d'Inventaires :** Le magasinier doit pouvoir **créer un nouvel inventaire**, en spécifiant une date (automatique par défaut, modifiable) et un statut initial (ex: "En cours"). Il doit pouvoir choisir d'inclure tous les articles ou une sélection spécifique.
- **RF8.2 : Saisie des Quantités Réelles :** Pour chaque article inclus dans un inventaire "En cours", le magasinier doit pouvoir **saisir la quantité physique réelle** trouvée en stock.
- **RF8.3 : Comparaison Stock Théorique vs. Réel :** Le système doit **calculer et afficher automatiquement l'écart** entre la quantité en stock théorique (calculée par le système à partir des mouvements) et la quantité réelle saisie par le magasinier.

- **RF8.4 : Affichage des Écarts :** Les **écarts (positifs ou négatifs) doivent être clairement visibles** pour chaque article inventorié.
- **RF8.5 : Ajustement du Stock :** Le magasinier doit pouvoir **valider l'inventaire**, ce qui aura pour effet d'**ajuster le stock théorique** de chaque article pour qu'il corresponde à la quantité réelle constatée. Une **raison d'ajustement** (ex: "Perte", "Casse", "Vol", "Erreur de saisie précédente") doit être **obligatoire** pour chaque ajustement.
- **RF8.6 : Historique des Inventaires :** Le système doit **conserver un historique de tous les inventaires** effectués, affichant la date, le statut (ex: "Terminé"), et un récapitulatif des écarts constatés.

3.9. Exportation des Données (RF9)

Le logiciel doit permettre d'exporter diverses données du système pour analyse externe ou archivage.

- **RF9.1 : Exportation des Inventaires :** Le magasinier doit pouvoir **exporter les données d'un inventaire spécifique** (date, articles, stock théorique, stock réel, écarts, ajustements) dans un format standard (ex: **CSV**, **Excel**).
- **RF9.2 : Exportation des Articles :** Le magasinier doit pouvoir **exporter la liste complète des articles** (nom, description, unité, stock actuel, seuil d'alerte, etc.) dans un format standard (ex: **CSV**, **Excel**).
- **RF9.3 : Exportation des Mouvements :** Le magasinier doit pouvoir **exporter l'historique des entrées et sorties** (avec filtres par période ou article) dans un format standard (ex: **CSV**, **Excel**).

4. Structure Détaillée du Projet

La conception de l'application MatChantier repose sur une architecture modulaire en Java natif, visant à garantir la robustesse, la maintenabilité et l'évolutivité. L'absence de base de données externe oriente la structure vers l'intégration d'une solution de persistance embarquée.

4.1. Architecture Générale

Le projet sera structuré en plusieurs couches principales, favorisant la séparation des préoccupations :

- **Couche Présentation (Interface Utilisateur - UI) :** Cette couche sera responsable de l'affichage de l'interface graphique et de la gestion des interactions utilisateur. Elle sera développée en JavaFX (ou Swing, selon le choix technologique spécifique), offrant une expérience utilisateur riche et native.
- **Couche Logique Métier (Business Logic) :** C'est le cœur de l'application, où les

règles métier définies dans les exigences fonctionnelles seront implémentées. Elle gérera la création, la modification, la suppression et la consultation des articles, les mouvements de stock, les inventaires, et les alertes. Cette couche sera indépendante de la couche de présentation et de la couche de données.
- **Couche Accès aux Données (Data Access Layer - DAL) :** Cette couche sera chargée de toutes les opérations d'interaction avec la base de données SQLite embarquée. Elle abstraira la logique d'accès aux données de la logique métier, permettant ainsi de changer de technologie de persistance si nécessaire sans impacter le reste de l'application.
- **Couche Utilitaires/Services :** Cette couche regroupera des fonctionnalités transversales telles que la gestion des fichiers d'exportation (CSV, PDF), des outils de validation, et d'autres services généraux.

4.2. Structure des Composants Clés

Chaque composant sera développé en Java et interagit de manière définie.

- **Modules d'Interface Utilisateur (UI Modules) :**
  - **Fenêtre Principale (Dashboard) :** Servira de point d'entrée pour le magasinier, affichant les indicateurs clés de performance (KPIs) et permettant une navigation vers les différentes sections de l'application.
  - **Gestion des Articles :** Formulaires pour la création et la modification d'articles, tableau pour la consultation.
  - **Gestion des Mouvements :** Formulaires dédiés aux entrées et sorties, et une vue historique des transactions.
  - **Gestion des Inventaires :** Interface guidée pour la création, la saisie des quantités physiques, l'affichage des écarts et la validation des ajustements.
  - **Génération de Rapports :** Interface pour la sélection des types de rapports et des options d'exportation.
  - **Module de Connexion/Authentification :** Interface pour la connexion et le changement de mot de passe du magasinier.
- **Classes de Logique Métier (Business Logic Classes) :**
  - **Gestionnaire d'Articles (ArticleManager) :** Encapsulera toutes les règles métier liées aux articles (validation, désactivation, etc.).
  - **Gestionnaire de Mouvements (MovementManager) :** Gérera l'enregistrement des entrées et sorties, et la mise à jour automatique des stocks.
  - **Gestionnaire d'Inventaires (InventoryManager) :** Orchestrera le processus d'inventaire, le calcul des écarts et l'application des ajustements.
  - **Gestionnaire d'Alertes (AlertManager) :** Surveillera les seuils de stock et déclenchera les alertes visuelles.

    - **Gestionnaire d'Utilisateurs (UserManager) :** Gérera l'authentification et les informations du magasinier.
- **Classes d'Accès aux Données (DAO - Data Access Objects) :**
    - **ArticleDAO :** Responsable des opérations CRUD (Create, Read, Update, Delete) sur la table articles.
    - **MovementDAO :** Responsable des opérations CRUD sur la table mouvements.
    - **InventoryDAO :** Gérera la persistance des données d'inventaire et des ajustements.
    - **UserDAO :** Gérera la persistance des informations d'authentification du magasinier.
    - **DatabaseConnectionManager :** Classe utilitaire pour établir et gérer la connexion à la base de données SQLite (inventaire_chantier.db). Elle sera également responsable de la création initiale des schémas de table si la base de données est vide.
- **Modèles de Données (Model Classes) :**
    - **Article :** Classe Java représentant un article avec ses attributs (id, nom, stock, minStock, etc.).
    - **Movement :** Classe Java représentant une transaction de mouvement (id, articleId, type, quantité, date, etc.).
    - **Inventory :** Classe Java représentant un inventaire (id, date, statut, liste d'articles inventoriés, etc.).
    - **User :** Classe Java pour les informations d'authentification.
- **Modules d'Exportation :**
    - **CSVExporter :** Classe utilitaire pour générer des fichiers CSV à partir des données extraites.
    - **PDFGenerator :** Classe utilitaire (nécessitera potentiellement une bibliothèque externe comme Apache PDFBox ou iText) pour générer des rapports PDF.

4.3. Stockage des Données (SQLite)

La base de données SQLite sera un fichier unique (inventaire_chantier.db) qui contiendra les tables suivantes :

- **articles :** Stockera les informations de base de chaque article (ID, Nom, Description, Unité, Catégorie, StockActuel, SeuilAlerte, PrixUnitaire, Actif).
- **mouvements :** Enregistrera chaque entrée et sortie (ID, ArticleID, TypeMouvement (Entrée/Sortie), Quantite, DateHeure, Fournisseur/ServiceAffectation, PersonneConcernee, Raison).
- **inventaires :** Contenra les métadonnées de chaque inventaire réalisé (ID,

DateInventaire, Statut).
- **inventaire_details :** Détaillera chaque article dans un inventaire spécifique (InventaireID, ArticleID, StockTheorique, StockPhysique, Ecart, RaisonAjustement, DateAjustement).

5. Guide d'Installation et d'Exécution du Programme

Ce guide fournit les étapes nécessaires pour installer et exécuter l'application MatChantier sur un système Windows avec un JRE compatible.

5.1. Pré-requis

- **Java Runtime Environment (JRE) :** La version 8 ou supérieure de JRE doit être installée sur la machine cible. Vous pouvez la télécharger depuis le site officiel d'Oracle ou d'OpenJDK.
    - Pour vérifier votre version de JRE, ouvrez l'Invite de Commande (CMD) et tapez : java -version
- **Java 17 :** La version de Java compatible avec le logiciel est la version 17. Le guide d’installation est fourni avec le document et le logiciel.

5.2. Installation de l'Application

L'application MatChantier sera distribuée sous la forme d'un fichier exécutable .jar (ou un installeur natif si un outil comme JLink/JPackage est utilisé).

1. **Téléchargement :** Obtenez le fichier MatChantier.jar (et potentiellement le fichier inventaire_chantier.db s'il est pré-rempli, sinon il sera créé au premier lancement).
2. **Emplacement :** Placez le fichier MatChantier.jar dans un dossier de votre choix sur votre ordinateur (par exemple, C:\Program Files\MatChantier\ ou C:\Users\VotreNom\Documents\MatChantier\). Il est recommandé de créer un dossier dédié.
3. **Base de Données :** Si le fichier inventaire_chantier.db n'est pas fourni avec le .jar, il sera automatiquement créé dans le même répertoire que le fichier .jar lors du premier lancement de l'application.

5.3. Exécution du Programme

5.3.1. Exécution via l'Explorateur de Fichiers (Recommandée pour l'utilisateur final)

1. **Naviguez** jusqu'au dossier où vous avez placé le fichier MatChantier.jar.
2. **Double-cliquez** simplement sur le fichier MatChantier.jar.
3. L'application devrait démarrer et afficher la fenêtre de connexion.

5.3.2. Exécution via l'Invite de Commande (Pour le débogage ou les utilisateurs

avancés)

1. Ouvrez l'**Invite de Commande** (tapez cmd dans la barre de recherche Windows et appuyez sur Entrée).
2. Naviguez jusqu'au répertoire où se trouve le fichier MatChantier.jar en utilisant la commande cd :
   cd C:\Users\VotreNom\Documents\MatChantier

   (Remplacez le chemin par l'emplacement réel de votre fichier).
3. Exécutez l'application en utilisant la commande java -jar :
   java -jar MatChantier.jar

4. L'application devrait démarrer. Tous les messages d'erreur ou de débogage seront affichés dans la fenêtre de l'Invite de Commande.