Nom : KABORE

Prenoms :BASNKOWINDE ALIX LARISSA

Nom : SECK

Prenoms : Cheick Omar Nouredine

Nom : OUEDRAOGO

Prenoms : Lidwine Axelle

UVBF/Génie Logiciel pure developer L2 2024

SUJET : `1. Système de Gestion d'Inventaire pour un Magasin Ajouter un nouvel article avec nom, catégorie, prix et quantité. Modifier les informations d'un article existant. Supprimer un article du stock. Rechercher des articles par nom Afficher tous les articles disponibles en stock avec leur quantité actuelle. Enregistrer les transactions (ajout et retrait d'inventaire) dans un fichier de log. Sauvegarder et charger l'état de l'inventaire dans un fichier pour maintenir la persistance des données.`

Rapport sur la Gestion des Produits et Ventes

1. Introduction

Dans le cadre de notre projet de développement d'un système de gestion de produits et de ventes, nous avons mis en place une application console en C++ qui permet à un utilisateur d'ajouter, modifier, supprimer et rechercher des produits. De plus, notre application permet de sauvegarder ces produits dans un fichier et de gérer les ventes effectuées. Le projet a été divisé en plusieurs parties que nous avons chacune abordées en équipe.

2. Répartition des Tâches

Le projet a été divisé en trois parties principales, et chaque membre du groupe a pris en charge une ou plusieurs fonctionnalités spécifiques du système :

- **Nouredine** : Responsable de la gestion des produits, de l'ajout, de la modification, de la suppression et de la recherche de produits. Il a également pris en charge la sauvegarde et le chargement des produits depuis/vers un fichier.
- **Larissa** : Responsable de l'enregistrement des ventes et de l'affichage des ventes. Elle a également contribué à la gestion de la lecture et de l'écriture dans les fichiers associés.
- **Axelle** : Responsable de la gestion générale du programme, de l'interface utilisateur, de la boucle principale permettant de choisir entre les différentes options et de la gestion des erreurs et des entrées utilisateur.

3. Implémentation

3.1. Gestion des Produits (Nouredine)

La gestion des produits est au cœur de notre système. Elle permet à l'utilisateur d'ajouter, de

modifier, de supprimer, et de rechercher des produits.

Ajout d'un produit : Nous avons utilisé un vecteur (`std::vector`) pour stocker la liste des produits. La fonction **`ajouterProduit()`** demande à l'utilisateur de saisir le nom, la description, le prix et la quantité du produit. Ensuite, le produit est ajouté au vecteur et les données sont sauvegardées dans un fichier **`produits.txt`** pour permettre une persistance des données.

```
void ajouterProduit() {
    Produit produit;

    std::cout << "Nom du produit : ";
    std::cin.ignore(); // Ignore les caractères résiduels
    std::getline(std::cin, produit.nom);

    std::cout << "Description : ";
    std::getline(std::cin, produit.description);

    std::cout << "Prix : ";
    while (!(std::cin >> produit.prix)) {
        std::cout << "Entrée invalide. Veuillez entrer un nombre pour le prix :
";
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    }

    std::cout << "Quantité : ";
    while (!(std::cin >> produit.quantite)) {
        std::cout << "Entrée invalide. Veuillez entrer un nombre entier pour la
quantité : ";
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    }

    produits.push_back(produit);
    sauvegarderProduits();  // Sauvegarde après ajout

    std::cout << "Produit ajouté avec succès !\n";
}
```

Modification et suppression de produits : La gestion de la modification et de la suppression repose sur la recherche du produit par son nom. Une fois le produit trouvé, l'utilisateur peut modifier les informations ou supprimer le produit de la liste.

```
// Exemple de modification
void modifierProduit() {
    std::string nom;
    std::cout << "Nom du produit à modifier : ";
    std::cin.ignore();
    std::getline(std::cin, nom);

    for (auto &produit : produits) {
        if (produit.nom == nom) {
            std::cout << "Modifier description : ";
            std::getline(std::cin, produit.description);
            std::cout << "Modifier prix : ";
            std::cin >> produit.prix;
            std::cout << "Modifier quantité : ";
            std::cin >> produit.quantite;
            sauvegarderProduits(); // Sauvegarde après modification
            std::cout << "Produit modifié avec succès !\n";
            return;
        }
    }
```

```
    std::cout << "Produit non trouvé.\n";
}
```

Chargement et sauvegarde des produits : Le chargement des produits est effectué en lisant le fichier **`produits.txt`**, et chaque produit est ajouté au vecteur **`produits`**. Les produits sont ensuite sauvegardés dans ce fichier après chaque modification, ajout ou suppression.

```
// Sauvegarde des produits
void sauvegarderProduits() {
    std::ofstream fichier("data/produits.txt", std::ios::trunc);
    if (fichier) {
        for (const auto &produit : produits) {
            fichier << produit.nom << "," << produit.description << ","
                    << produit.prix << "," << produit.quantite << "\n";
        }
        fichier.close();
        std::cout << "Produits sauvegardés avec succès !\n";
    } else {
        std::cerr << "Erreur : Impossible d'ouvrir le fichier produits.txt\n";
    }
}
```

3.2. Gestion des Ventes (Larissa)

La gestion des ventes permet d'enregistrer chaque vente réalisée, y compris les détails comme l'ID de la vente, le produit vendu, la quantité et la date. Les ventes sont ensuite sauvegardées dans un fichier **`ventes.txt`** pour la consultation future.

```
void enregistrerVente() {
    int id;
    std::string nomProduit;
    int quantiteVendue;
    std::string date;

    std::cout << "ID de la vente : ";
    std::cin >> id;
    std::cout << "Nom du produit : ";
    std::cin.ignore();
    std::getline(std::cin, nomProduit);
    std::cout << "Quantité vendue : ";
    std::cin >> quantiteVendue;
    std::cout << "Date de la vente (format YYYY-MM-DD) : ";
    std::cin >> date;

    std::ofstream fichier("data/ventes.txt", std::ios::app);
    if (fichier) {
        fichier << id << "," << nomProduit << "," << quantiteVendue << "," <<
date << "\n";
        fichier.close();
        std::cout << "Vente enregistrée avec succès !\n";
    } else {
        std::cerr << "Erreur : Impossible d'ouvrir le fichier ventes.txt\n";
    }
}
```

Liste des ventes : La fonction **`listerVentes()`** permet de lire le fichier **`ventes.txt`** et d'afficher toutes les ventes enregistrées.

```
void listerVentes() {
    std::ifstream fichier("data/ventes.txt");
```

```
    if (fichier) {
        std::string ligne;
        std::cout << "\n--- Liste des ventes ---\n";
        while (std::getline(fichier, ligne)) {
            std::cout << ligne << "\n";
        }
        fichier.close();
    } else {
        std::cerr << "Erreur : Impossible d'ouvrir le fichier ventes.txt\n";
    }
}
```

3.3. Interface Utilisateur et Gestion des Erreurs (Axelle)

La gestion de l'interface utilisateur et des erreurs a été implémentée par **Axelle**. elle a créé la boucle principale du programme qui permet à l'utilisateur de choisir une option parmi plusieurs, et chaque option correspond à une fonction du programme.

Nous avons également pris soin de gérer les erreurs d'entrée de manière appropriée, en utilisant **`std::cin.clear()`** et **`std::cin.ignore()`** pour garantir que l'utilisateur puisse entrer des données valides.

```
int main() {
    int choix;

    chargerProduits();  // Charger les produits au démarrage

    do {
        std::cout << "\n--- Gestion des produits ---\n";
        std::cout << "1. Ajouter un produit\n";
        std::cout << "2. Modifier un produit\n";
        std::cout << "3. Supprimer un produit\n";
        std::cout << "4. Rechercher un produit\n";
        std::cout << "5. Enregistrer une vente\n";
        std::cout << "6. Lister les ventes\n";
        std::cout << "7. Quitter\n";
        std::cout << "Choix : ";
        std::cin >> choix;

        switch (choix) {
            case 1:
                ajouterProduit();
                break;
            case 2:
                modifierProduit();
                break;
            case 3:
                supprimerProduit();
                break;
            case 4:
                rechercherProduit();
                break;
            case 5:
                enregistrerVente();
                break;
            case 6:
                listerVentes();
                break;
            case 7:
                std::cout << "Au revoir !\n";
                break;
```

```
            default:
                std::cout << "Choix invalide. Essayez à nouveau.\n";
        }
    } while (choix != 7);

    return 0;
}
```

4. Conclusion

Nous avons réussi à implémenter un système complet de gestion de produits et de ventes, avec une interface utilisateur simple et des fonctionnalités robustes pour l'ajout, la modification, la suppression, et la recherche des produits, ainsi que pour l'enregistrement et la consultation des ventes. Le programme utilise des fichiers pour sauvegarder les données, garantissant ainsi leur persistance.

Le projet a été une expérience enrichissante, nous permettant de mieux comprendre les principes de la gestion des données et de la programmation orientée objet en C++, ainsi que les défis de l'interface utilisateur et de la gestion des erreurs.

NB : Taper g++ -g src/main.cpp src/gestionProduits.cpp -o main pour compiler les fichier source et e**xécuter l'exécutable généré** avec `./main (il ya déjà un executable dans le zip mais il peut y avoir un probleme quand on le lance , donc pour eviter les desagraments vaut mieux supprimer et recompiler l’executable`)