Contexte et approche

Le rendu avec Massinissa a été un peu compliqué cette semaine. Nos emplois du temps étaient tellement différents qu'on n'a pas réussi à trouver un moment pour bosser ensemble sur le rendu.

Notre approche s'est focalisée sur l'implémentation fonctionnelle des exigences demandées :

1. Support de deux formats de stockage : JSON (par défaut) et CSV (nouvelle fonctionnalité)
2. Modification des commandes de base : add, remove, list
3. Implémentation de la commande `total`

Choix techniques et architecture

J'ai choisi de créer une classe `GroceryItem` pour représenter les articles, ce qui apporte plusieurs avantages :

- Modularité et peut nous aider sur l'évolution du code sur la durée
- Séparation claire des responsabilités
- Possibilité d'enrichir facilement les fonctionnalités des articles

Pour le format JSON, j'ai opté pour une liste d'objets GroceryItem, en utilisant Jackson pour la sérialisation/désérialisation. Pour le format CSV, j'ai utilisé Apache Commons CSV avec une structure simple :

- En-tête avec noms des colonnes (name, quantity)
- Valeurs séparées par des virgules
- Un article par ligne

État actuel et limites

Le code actuel remplit les exigences fonctionnelles mais présente certaines limites :

- Il ne respecte pas encore pleinement les principes SOLID
- Une factorisation sera nécessaire pour améliorer la maintenabilité

J'ai fait le choix de privilégier le fonctionnement correct des commandes et la prise en charge des deux formats de fichier, tout en sachant que des améliorations architecturales seraient nécessaires par la suite.

Prochaines étapes

Pour les prochaines itérations, on prévoit :

1. Développer une suite complète de tests unitaires
2. Créer une interface dédiée à la gestion des formats de fichier pour mieux respecter le principe d'ouverture/fermeture
3. Refactoriser le code pour une meilleure séparation des responsabilités
4. Améliorer la gestion des erreurs et l'expérience utilisateur

Ce premier TP m'a permis de poser les bases fonctionnelles. L'objectif est maintenant d'améliorer progressivement la qualité du code en respectant le principe SOLID tout en ajoutant de nouvelles fonctionnalités selon les besoins du projet.