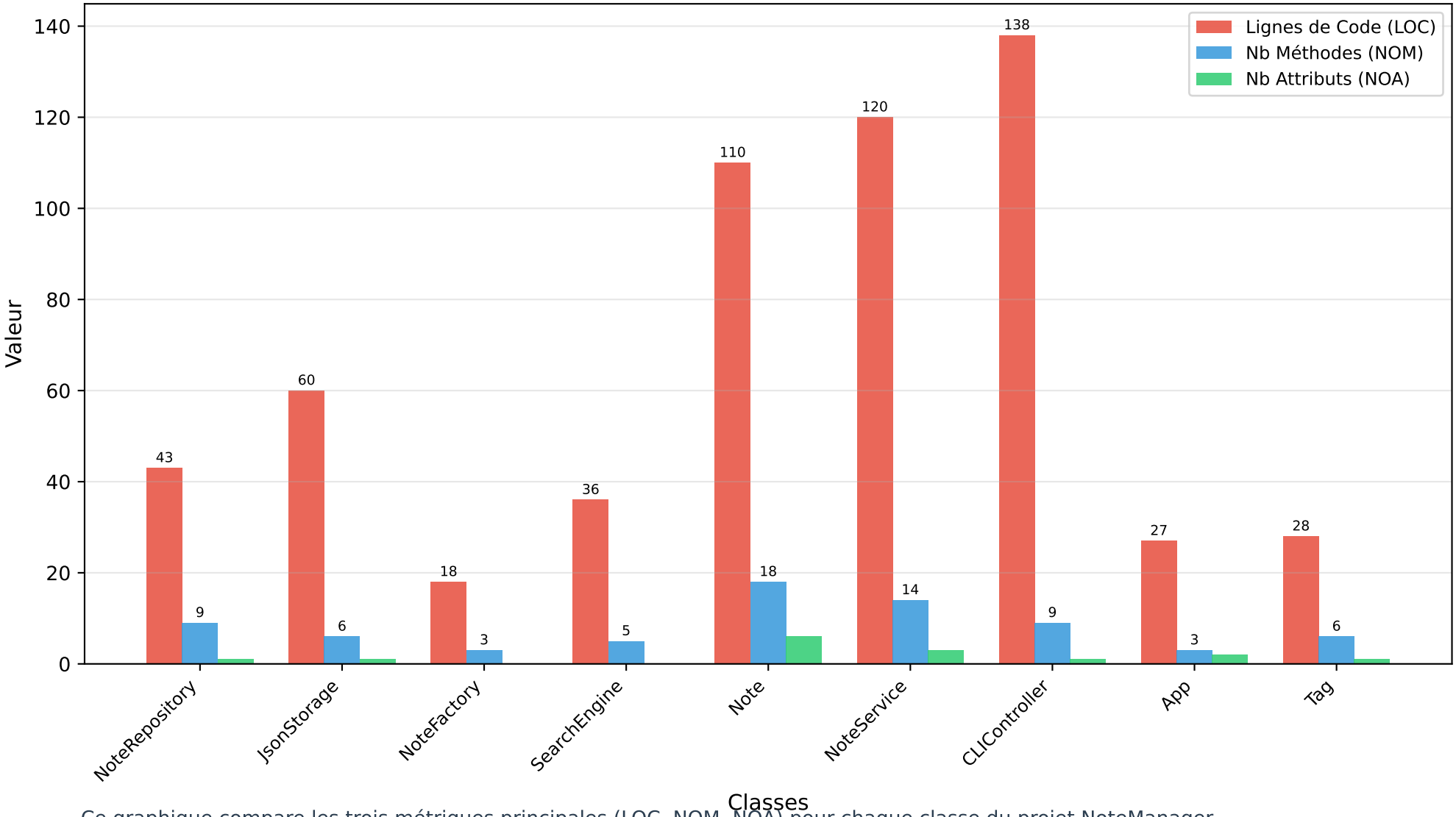
Rapport de Métriques de Qualité

Projet NoteManager — TP2 MGL843

Analyse automatisée par pipeline CI/CD

Généré le 2026-02-16 à 02:20

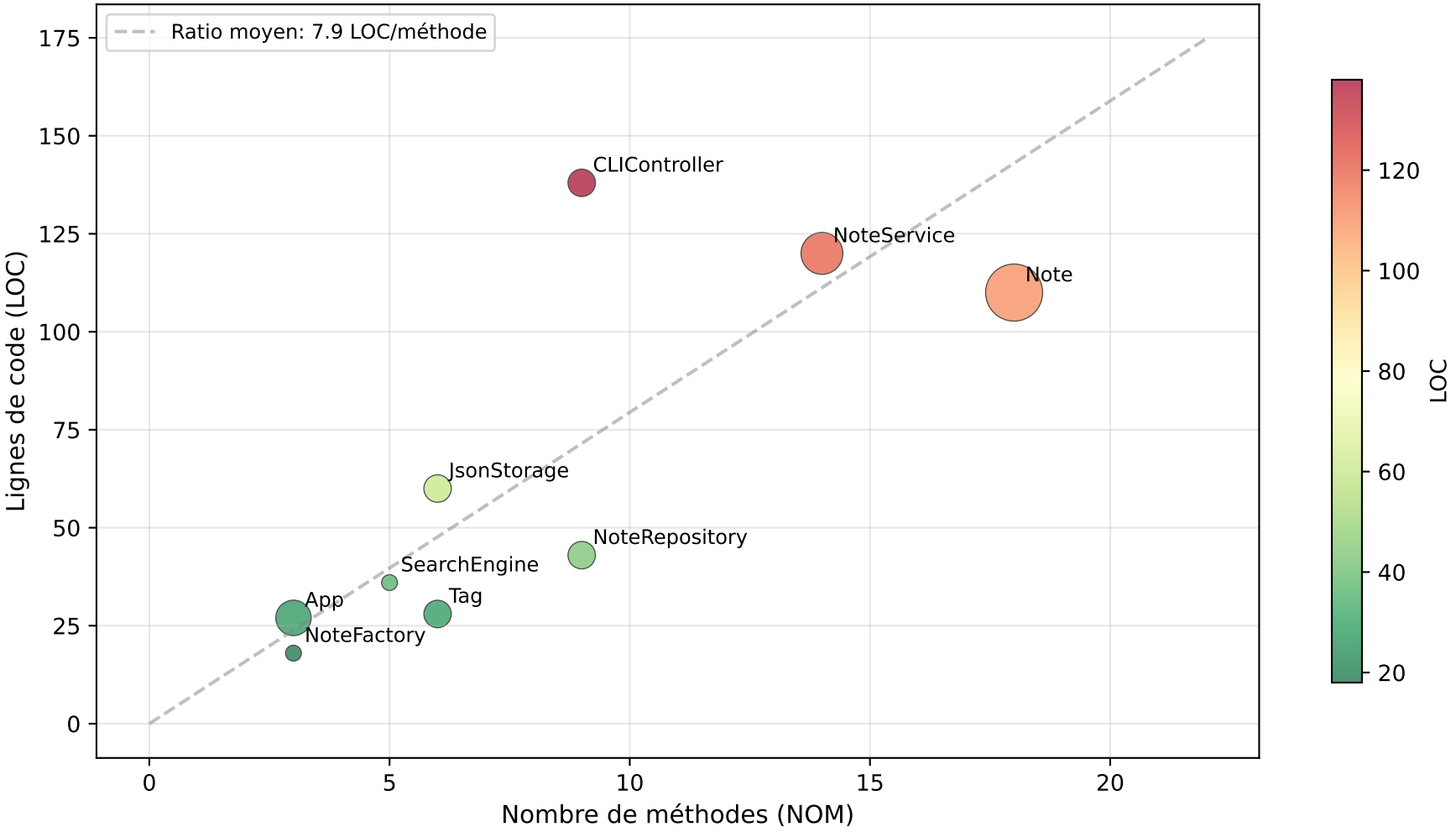
1. Histogramme des métriques par classe



Ce graphique compare les trois métriques principales (LOC, NOM, NOA) pour chaque classe du projet NoteManager.

La classe CLIController domine en termes de lignes de code (138 LOC), tandis que Note possède le plus grand nombre de méthodes (18). Un déséquilibre entre LOC et NOM peut indiquer des méthodes trop longues ou un manque de décomposition fonctionnelle. Les classes avec un LOC élevé et peu de méthodes sont des candidates au refactoring.

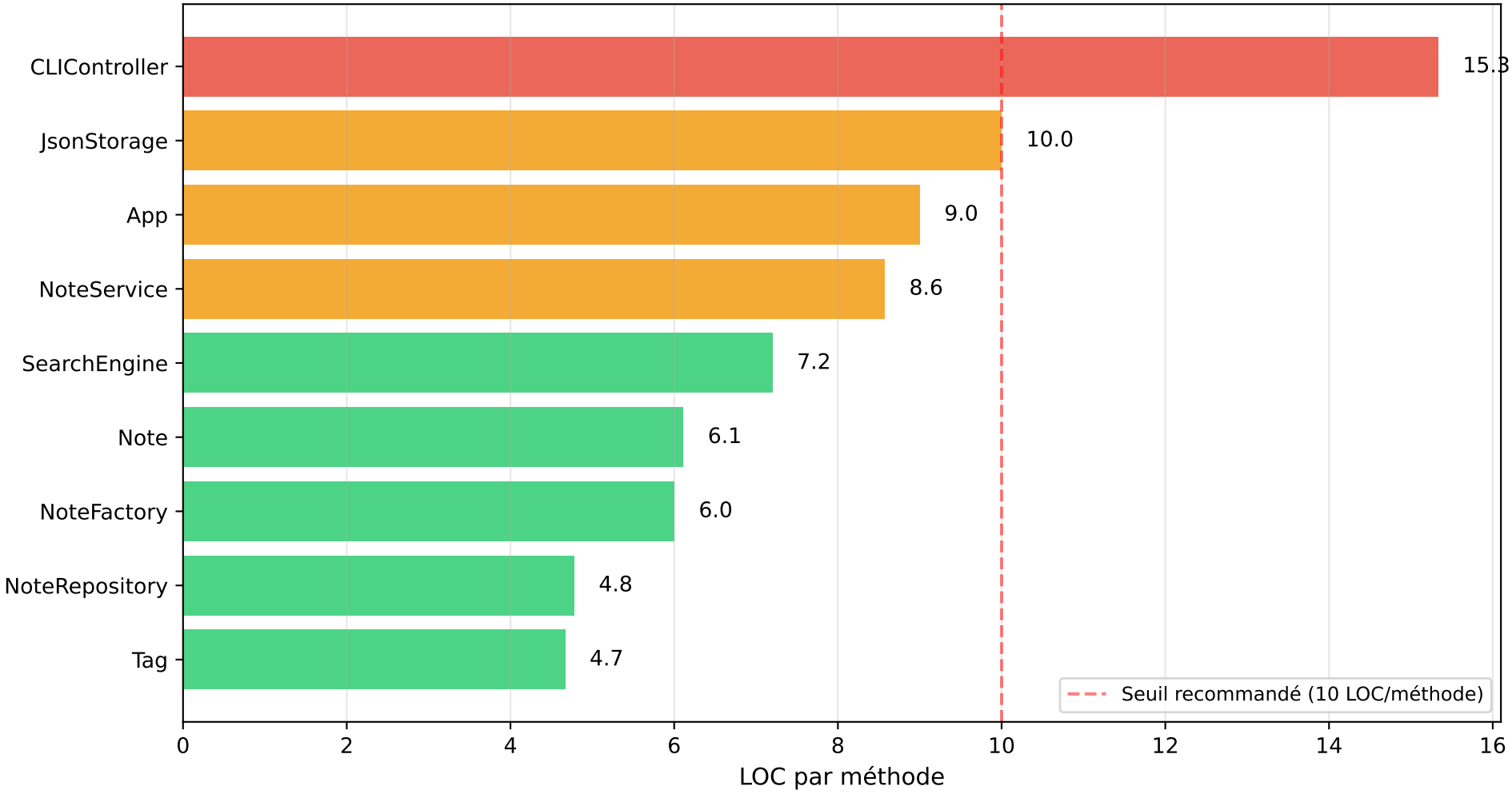
2. Diagramme de dispersion — LOC vs NOM



Ce diagramme positionne chaque classe selon son nombre de méthodes (axe X) et ses lignes de code (axe Y). La taille des cercles est proportionnelle au nombre d'attributs (NOA). La ligne pointillée représente le ratio moyen de 7.9 LOC par méthode.

Classes au-dessus de la moyenne (méthodes plus longues) : JsonStorage, NoteService, CLIController, App.
Classes en dessous (méthodes plus concises) : NoteRepository, NoteFactory, SearchEngine, Note, Tag.
Les classes éloignées de la ligne de tendance méritent une attention particulière lors du refactoring.

3. Densité de code par classe (LOC / méthode)



Ce graphique montre le ratio LOC/méthode pour chaque classe, trié par ordre croissant. Il permet d'identifier les classes dont les méthodes sont trop longues.

Vert = bon (< 8 LOC/méthode), Orange = acceptable (8-12), Rouge = à surveiller (> 12).
La densité moyenne du projet est de 8.0 LOC/méthode. Les classes dépassant le seuil de 10 sont : CLIController. Elles pourraient bénéficier d'une décomposition en sous-méthodes.

4. Tableau récapitulatif des métriques

Classe	NOM	NOA	LOC	LOC/Méthode
NoteRepository	9	1	43	4.8
JsonStorage	6	1	60	10.0
NoteFactory	3	0	18	6.0
SearchEngine	5	0	36	7.2
Note	18	6	110	6.1
NoteService	14	3	120	8.6
CLIController	9	1	138	15.3
App	3	2	27	9.0
Tag	6	1	28	4.7

Ce tableau présente l'ensemble des métriques collectées pour les 9 classes du projet.

Statistiques globales du projet :
- Total LOC : 580 | Total méthodes : 73 | Total attributs : 15
- Ratio moyen LOC/méthode : 7.9
- Classe la plus volumineuse : CLIController (138 LOC)
- Classe avec le plus de méthodes : Note (18 méthodes)

Conclusion et recommandations

Vue d'ensemble

Le projet NoteManager est composé de 9 classes, totalisant 580 lignes de code réparties dans 73 méthodes. La densité moyenne est de 8.0 LOC par méthode.

Points forts

- 5 classe(s) avec une bonne densité (< 8 LOC/méthode) : NoteRepository, NoteFactory, SearchEngine, Note, Tag.
  Ces classes montrent une bonne décomposition fonctionnelle.

Points d'attention

- 1 classe(s) à risque élevé (> 12 LOC/méthode) : CLIController.
  Recommandation : décomposer les méthodes longues en sous-méthodes.
- 3 classe(s) à surveiller (8-12 LOC/méthode) : JsonStorage, NoteService, App.

Classe la plus volumineuse : CLIController (138 LOC, 9 méthodes). Cette classe concentre 24% du code total et pourrait bénéficier d'une répartition des responsabilités.

Pipeline de validation

Ces métriques ont été générées automatiquement par un pipeline CI/CD GitHub Actions, puis validées par vérification croisée entre Pharo/Moose et Python (4 niveaux de validation : structure, invariants, calcul indépendant, comparaison).