Projet Mesure Qualité et Performance Logicielle

M. Diouf

Licence 3 en Informatique (Génie Logiciel)

AWA KA

Ndeye Coumba Ndiaye

Rapport de test, mesure et qualité

Les commandes utilisées pour faire la mesure et évaluer la qualité du code :

I. flake8 :

Installation : pip install flake8

Cette commande vérifie la conformité avec la PEP8, relevant toutes les infractions trouvées dans les fichiers de code (c'est-à-dire les mauvais espacements, les lignes trop longues). Elle a permis de valider le code.

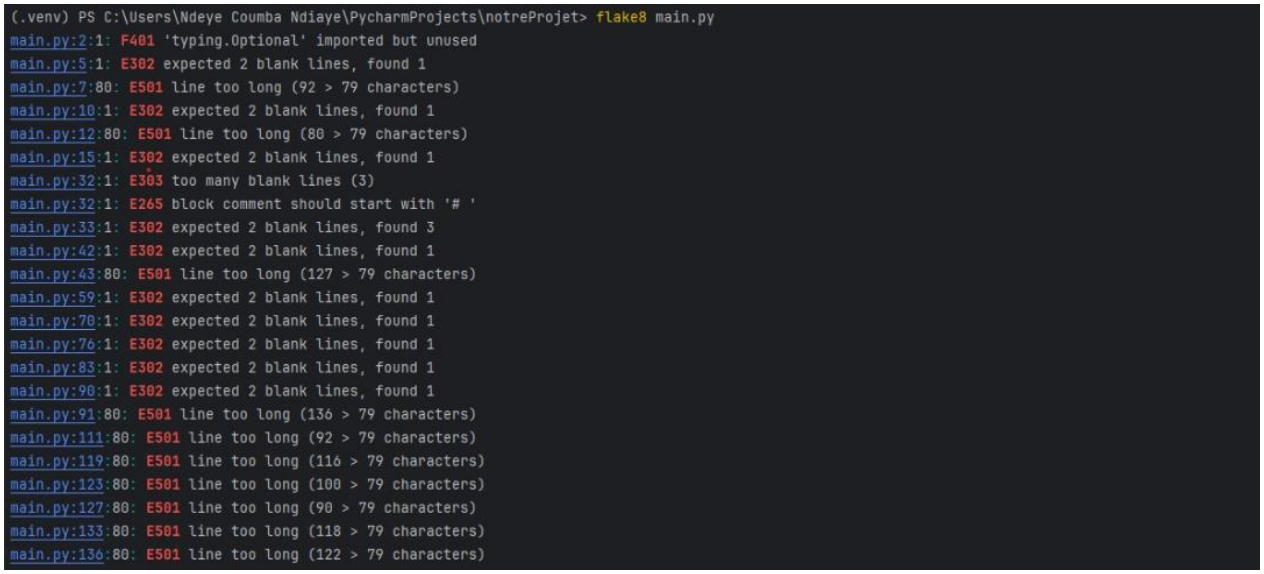

Figure 1 : Capture Résultats de la commande flake8 :

```
main.py:91:80: E501 line too long (136 > 79 characters)
main.py:111:80: E501 line too long (92 > 79 characters)
main.py:119:80: E501 line too long (116 > 79 characters)
main.py:123:80: E501 line too long (100 > 79 characters)
main.py:127:80: E501 line too long (90 > 79 characters)
main.py:133:80: E501 line too long (118 > 79 characters)
main.py:136:80: E501 line too long (122 > 79 characters)
main.py:137:80: E501 line too long (134 > 79 characters)
main.py:138:80: E501 line too long (100 > 79 characters)
main.py:142:80: E501 line too long (107 > 79 characters)
main.py:149:1: E305 expected 2 blank lines after class or function definition, found 1
main.py:166:11: E225 missing whitespace around operator
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet>
```

Suite Résultats de la commande flake8

II. Pylint

Installation : pip install pylint

Cette commande détecte les erreurs de programmation. C'est un outil qui permet d'identifier les erreurs avant l'exécution de notre programme. Pylint nous a permis de faire une analyse de notre code

```
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet> pylint main.py
************* Module main
main.py:174:0: C0301: Line too long (122/100) (line-too-long)
main.py:175:0: C0301: Line too long (134/100) (line-too-long)
main.py:180:0: C0301: Line too long (107/100) (line-too-long)
main.py:1:0: C0114: Missing module docstring (missing-module-docstring)
main.py:6:0: C0115: Missing class docstring (missing-class-docstring)
main.py:7:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:6:0: R0903: Too few public methods (1/2) (too-few-public-methods)
main.py:14:0: C0115: Missing class docstring (missing-class-docstring)
main.py:14:0: R0903: Too few public methods (1/2) (too-few-public-methods)
main.py:20:0: C0115: Missing class docstring (missing-class-docstring)
main.py:20:0: R0903: Too few public methods (1/2) (too-few-public-methods)
main.py:26:0: C0115: Missing class docstring (missing-class-docstring)
main.py:30:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:26:0: R0903: Too few public methods (1/2) (too-few-public-methods)
main.py:37:0: C0115: Missing class docstring (missing-class-docstring)
main.py:37:0: R0903: Too few public methods (1/2) (too-few-public-methods)
main.py:47:0: C0115: Missing class docstring (missing-class-docstring)
main.py:48:4: R0913: Too many arguments (7/5) (too-many-arguments)
main.py:65:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:68:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:73:0: C0115: Missing class docstring (missing-class-docstring)
main.py:77:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:80:4: C0116: Missing function or method docstring (missing-function-docstring)
```

Figure 2 : **Capture Résultats de la commande pylint :**

```
main.py:85:0: R0903: Too few public methods (0/2) (too-few-public-methods)
main.py:92:0: C0115: Missing class docstring (missing-class-docstring)
main.py:92:0: R0903: Too few public methods (0/2) (too-few-public-methods)
main.py:100:0: C0115: Missing class docstring (missing-class-docstring)
main.py:100:0: R0903: Too few public methods (0/2) (too-few-public-methods)
main.py:108:0: C0115: Missing class docstring (missing-class-docstring)
main.py:108:0: R0902: Too many instance attributes (13/7) (too-many-instance-attributes)
main.py:109:4: R0913: Too many arguments (6/5) (too-many-arguments)
main.py:115:8: W0621: Redefining name 'notification_context' from outer scope (line 190) (redefined-outer-name)
main.py:131:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:134:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:140:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:144:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:151:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:158:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:164:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:173:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:174:8: W0621: Redefining name 'rapport' from outer scope (line 254) (redefined-outer-name)
main.py:179:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:183:4: C0116: Missing function or method docstring (missing-function-docstring)
main.py:2:0: W0611: Unused Optional imported from typing (unused-import)

-----------------------------------
Your code has been rated at 6.48/10

(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet>
```

Suite Résultats de la commande pylint :

III. Mypy

Installation : pip install mypy

Cette commande vérifie le typage statique et détecte les erreurs de typage.
Mypy est capable de vérifier si les fonctions ont été déclarées avec le mauvais type de sortie.
Voici les résultats de cette commande nous montrant les erreurs de typage dans notre code

```
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet> mypy main.py
main.py:50: error: Need type annotation for "dependances" (hint: "dependances: list[<type>] = ...")  [var-annotated]
main.py:96: error: Need type annotation for "taches" (hint: "taches: list[<type>] = ...")  [var-annotated]
main.py:99: error: Need type annotation for "risques" (hint: "risques: list[<type>] = ...")  [var-annotated]
main.py:100: error: Need type annotation for "jalons" (hint: "jalons: list[<type>] = ...")  [var-annotated]
main.py:102: error: Need type annotation for "changements" (hint: "changements: list[<type>] = ...")  [var-annotated]
main.py:103: error: Need type annotation for "chemin_critique" (hint: "chemin_critique: list[<type>] = ...")  [var-annotated]
main.py:130: error: Argument 2 to "Changement" has incompatible type "float"; expected "int"  [arg-type]
Found 7 errors in 1 file (checked 1 source file)
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet>
```

Figure 3 : Capture Résultats de la commande mypy :

IV. Coverage

Installation : pip install coverage

Cette commande nous a permet de vérifier si la classe test existe dans la code et combien de test nous avons fait.
La capture ci-dessous nous montre les résultats de la commande

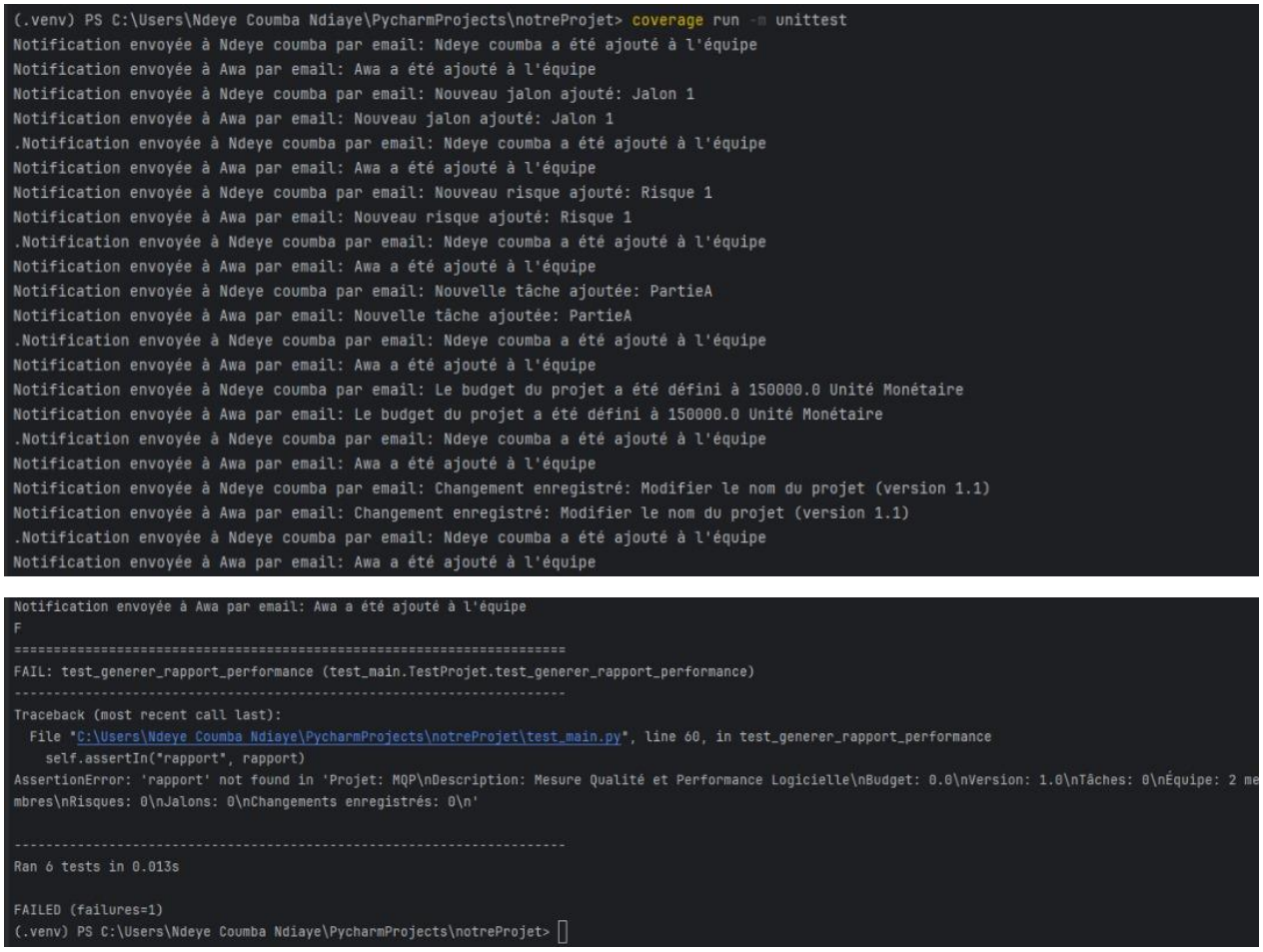

Figure 4 : Capture Résultats de la Commande coverage :

V. Vulutre

Installation : pip install vulture

Cette commande détecte les classes, la méthode, les imports inutilisés ou s'ils ont été utilisés partiellement.
La commande nous a permis de voir tous les méthodes et classe non utilisées
On a ici les résultats de la commande

```
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet> vulture main.py
main.py:2: unused import 'Optional' (90% confidence)
main.py:15: unused class 'SMSNotificationStrategy' (60% confidence)
main.py:55: unused method 'mettre_a_jour_statut' (60% confidence)
main.py:103: unused attribute 'chemin_critique' (60% confidence)
main.py:106: unused method 'set_notification_strategy' (60% confidence)
main.py:141: unused method 'calculer_chemin_critique' (60% confidence)
main.py:143: unused attribute 'chemin_critique' (60% confidence)
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet>
```

Figure 5 : Capture Résultats de la Commande vulture :

VI. Black

Installation : pip install black

Blake est un formateur code python. L'utilisation de cette commande nous a permis de reformater notre code comme le montre le résultats ci-dessous.

```
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet> black main.py
reformatted main.py

All done! ✨🍰 ✨
1 file reformatted.
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet>
```

Figure 6 : Capture Résultats de la Commande Blake :

VII. Radon

Installation : pip install radon

Le radon est un outil pour python qui calcule diverses mesures à partir du code source. Le radon permet de calculer la complexité de McCabe c'est-à-dire la complexité cyclomatique, Métriques brutes (il s'agit de SLOC, lignes de commentaires, lignes vides, etc.), Métrique Halstead, Index de maintenabilité
Voici les différentes commandes de radon :

1. **radon hal main.py**

```
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet> radon hal main.py
main.py:
    h1: 2
    h2: 7
    N1: 4
    N2: 8
    vocabulary: 9
    length: 12
    calculated_length: 21.651484454403228
    volume: 38.03910001730775
    difficulty: 1.1428571428571428
    effort: 43.47325716263743
    time: 2.415180953479857
    bugs: 0.012679700005769252
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet>
```

Figure 7 : Capture Résultats de la Commande radon hall :

2. Radon cc main.py

```
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet> radon cc main.py
main.py
    C 21:0 NotificationContext - A
    M 25:4 NotificationContext.notify - A
    C 5:0 NotificationStrategy - A
    C 10:0 EmailNotificationStrategy - A
    C 15:0 SMSNotificationStrategy - A
    C 33:0 Membre - A
    C 42:0 Tache - A
    C 59:0 Equipe - A
    C 70:0 Jalon - A
    C 76:0 Risque - A
    C 83:0 Changement - A
    C 90:0 Projet - A
    M 6:4 NotificationStrategy.envoyer - A
    M 11:4 EmailNotificationStrategy.envoyer - A
    M 16:4 SMSNotificationStrategy.envoyer - A
    M 22:4 NotificationContext.__init__ - A
    M 34:4 Membre.__init__ - A
    M 38:4 Membre.__str__ - A
    M 43:4 Tache.__init__ - A
    M 52:4 Tache.ajouter_dependance - A
    M 55:4 Tache.mettre_a_jour_statut - A
    M 60:4 Equipe.__init__ - A
    M 63:4 Equipe.ajouter_membre - A
```

```
    M 22:4 NotificationContext.__init__ - A
    M 34:4 Membre.__init__ - A
    M 38:4 Membre.__str__ - A
    M 43:4 Tache.__init__ - A
    M 52:4 Tache.ajouter_dependance - A
    M 55:4 Tache.mettre_a_jour_statut - A
    M 60:4 Equipe.__init__ - A
    M 63:4 Equipe.ajouter_membre - A
    M 66:4 Equipe.obtenir_membres - A
    M 71:4 Jalon.__init__ - A
    M 77:4 Risque.__init__ - A
    M 84:4 Changement.__init__ - A
    M 91:4 Projet.__init__ - A
    M 106:4 Projet.set_notification_strategy - A
    M 109:4 Projet.ajouter_tache - A
    M 113:4 Projet.ajouter_membre_equipe - A
    M 117:4 Projet.definir_budget - A
    M 121:4 Projet.ajouter_risque - A
    M 125:4 Projet.ajouter_jalon - A
    M 129:4 Projet.enregistrer_changement - A
    M 135:4 Projet.generer_rapport_performance - A
    M 141:4 Projet.calculer_chemin_critique - A
    M 145:4 Projet.notifier - A
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet>
```

Figure 8 : Capture Résultats de la Commande radon cc :

3. Radon raw main.py

```
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet> radon  raw main.py
main.py
    LOC: 220
    LLOC: 128
    SLOC: 155
    Comments: 23
    Single comments: 22
    Multi: 0
    Blank: 43
    - Comment Stats
        (C % L): 10%
        (C % S): 15%
        (C + M % L): 10%
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet>
```

Figure 9 : Capture Résultats de la Commande radon raw :

VIII. Pyflakes

Installation : pip install pyflakes

Cette commande analyse les programmes et détecte diverses erreurs comme les erreurs de syntaxe et les problèmes de style. Il fonctionne en analysant le fichier source.

```
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet> pyflakes main.py
main.py:2:1: 'typing.Optional' imported but unused
(.venv) PS C:\Users\Ndeye Coumba Ndiaye\PycharmProjects\notreProjet>
```

Figure 10 : Capture Résultats de la commande flakes