



**UNIVERSITE IBA DER THIAM DE THIES**  
**U.F.R DES SCIENCES ET TECHNOLOGIES**  
**DEPARTEMENT D'INFORMATIQUE**

Projet Mesure Qualité et Performance Logicielle  
Licence 3 Informatique (Génie Logiciel)  
2022-2023

Membres \_\_\_\_\_ :

**Thierno Adama WADE**

**Ibrahima Fall**

**Seydina Ababacar NDOUR**

**Rapport :**

Ce rapport de projet consiste à détailler et à clarifier les tâches effectuées sur nos choix et réalisations dans notre processus de développement pour une meilleure compréhension de notre travail.

Dans ce rapport, l'objectif est de mettre en exergue l'influence ainsi que l'importance que peuvent avoir les outils d'analyse, de mesure et qualité de code dans le langage Python.

## **Partie B** : Test, Mesure et Qualité du code

Cette partie nous renseigne sur la description des exigences du modèle à implémenter. Après *rédaction de la classe de test (unittest) et exécution des tests*, nous allons passer vers l'*analyse du code avec les mesures de qualités*.

### **I) Explication du Test**

On a utilisé un ensemble de fonctions qui permettent de tester le code principal. Elles sont les suivantes :

1. *test\_ajouter\_membre* : Cette fonction est utilisée pour vérifier que les membres Ibrahima et Thierno Adama ont été bien ajoutés à l'équipe du projet.

- **Résultat attendu** : 2 membres, avec les noms correspondants.

2. *test\_ajouter\_tache* : On a utilisé cette fonction pour vérifier si les tâches "Développer l'API" et "Intégration des fonctionnalités" ont été ajoutées au projet.

- **Résultat attendu** : 2 tâches, avec les noms correspondants.

3. *test\_ajouter\_risque* : Elle permet de tester que si le risque "Retard possible dû à des dépendances externes" a été bien ajouté au projet.

- **Résultat attendu** : 1 risque, avec la description correspondante.

4. *test\_ajouter\_jalon* : Elle va vérifier si le jalon "Phase 1 achevée" a été ajouté au projet.

- **Résultat attendu** : 1 jalon, avec le nom correspondant.

5. *test\_enregistrer\_changement* : Cette fonction va tester que si le changement "Changement de la version de l'API" a été enregistré dans le projet.

- **Résultat attendu** : 1 changement, avec la description correspondante.

6. *test\_chemin\_critique* : Cette fonction permet de vérifier si le calcul du chemin critique renvoie les tâches correctes.

- **Résultat attendu** : Le chemin critique doit inclure les tâches "Développer l'API" et "Intégration des fonctionnalités".

7. *test\_generer\_rapport\_activite* : Vérifie que le rapport d'activité généré contient les informations attendues.

- **Résultat attendu** : Le rapport doit contenir les noms des membres de l'équipe, les tâches, les risques, les jalons et les changements.

## II) Mesure et Qualité du code

L'analyse est prévue pour les fichiers à savoir ***classes\_principales.py*** et ***phase\_test.py***.

Voici les outils que l'on a eu à utiliser lors de la conception après installation. Nous allons illustrer les faits ainsi que les tâches effectuées par ceux-là :

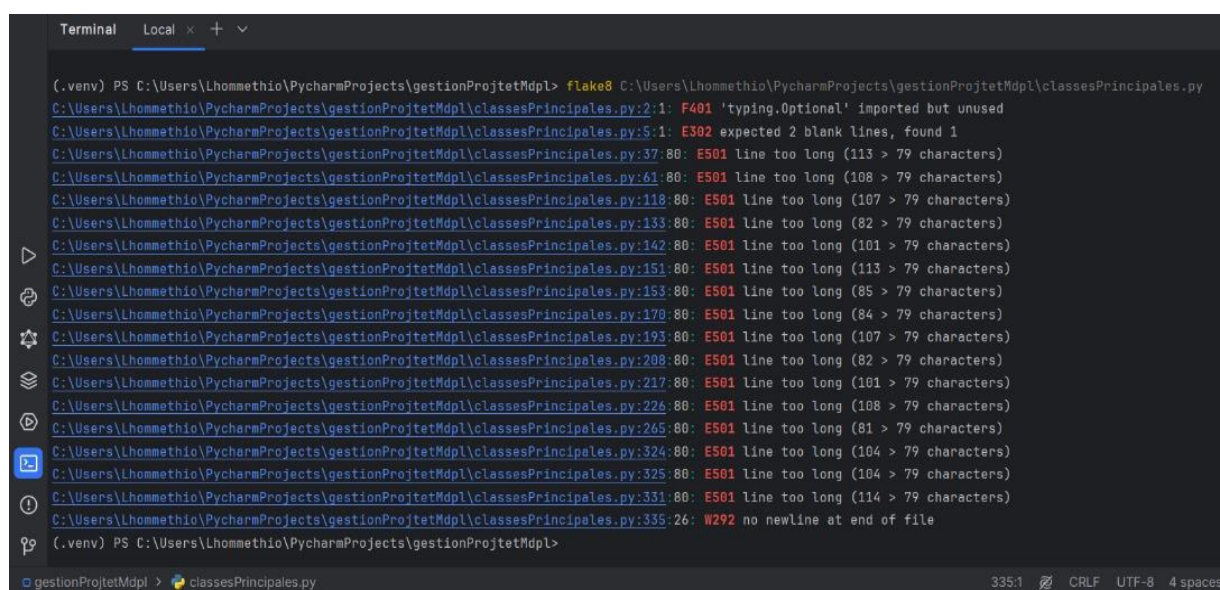
- flake8
- pylint
- mypy
- coverage
- vulture
- Black
- radon
- pyflakes

### 1. ***classes\_principales.py*** :

Nous allons illustrer les résultats de l'analyse de notre code avec les différents outils de mesure de qualité du code :

- **flake8**

Après avoir exécuté la commande au niveau de notre fichier, nous avons eu ceci comme première impression :



```
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl> flake8 C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:2:1: F401 'typing.Optional' imported but unused
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:5:1: E302 expected 2 blank lines, found 1
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:37:80: E501 line too long (113 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:61:80: E501 line too long (108 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:118:80: E501 line too long (107 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:133:80: E501 line too long (82 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:142:80: E501 line too long (101 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:151:80: E501 line too long (113 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:153:80: E501 line too long (85 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:178:80: E501 line too long (84 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:193:80: E501 line too long (107 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:208:80: E501 line too long (82 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:217:80: E501 line too long (101 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:226:80: E501 line too long (108 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:265:80: E501 line too long (81 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:324:80: E501 line too long (104 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:325:80: E501 line too long (104 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:331:80: E501 line too long (114 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classesPrincipales.py:335:26: W292 no newline at end of file
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl>
```

On note que plusieurs des lignes signalent que les nombres de caractères dépassent la règle E501 qui stipule que les lignes de code ne doivent pas dépasser 79 caractères.

De par cela, nous avons tenté d'effectuer des améliorations pour celle-ci par cette illustration :

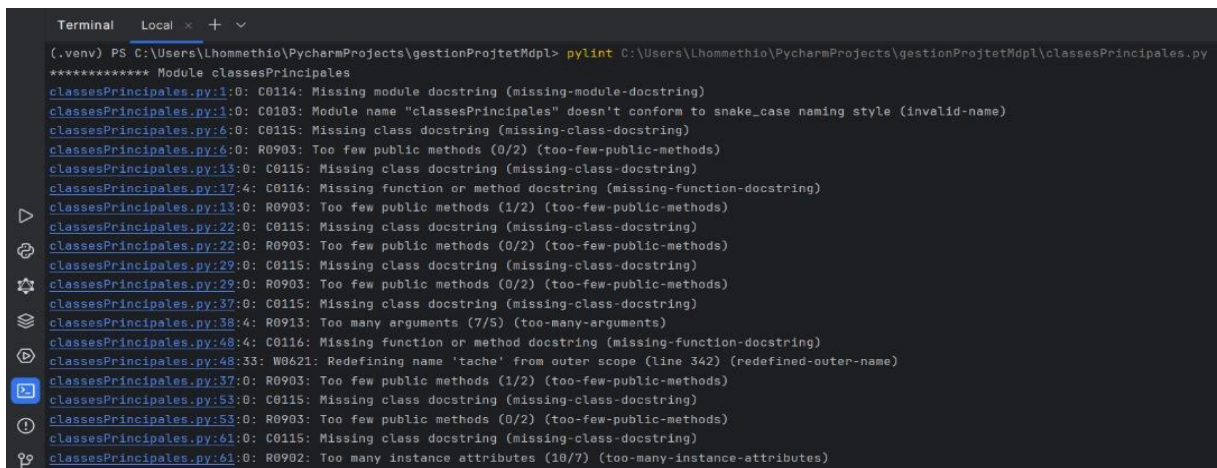


```
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> flake8 C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:2:1: F401 'typing.Optional' imported but unused
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:336:80: E501 line too long (81 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:337:80: E501 line too long (81 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:343:80: E501 line too long (114 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:347:26: W292 no newline at end of file
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> flake8 C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:2:1: F401 'typing.Optional' imported but unused
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:343:80: E501 line too long (114 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:347:26: W292 no newline at end of file
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> flake8 C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:2:1: F401 'typing.Optional' imported but unused
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:344:80: E501 line too long (87 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:348:26: W292 no newline at end of file
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> flake8 C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:2:1: F401 'typing.Optional' imported but unused
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py:348:26: W292 no newline at end of file
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> flake8 C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> flake8 C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp>
```

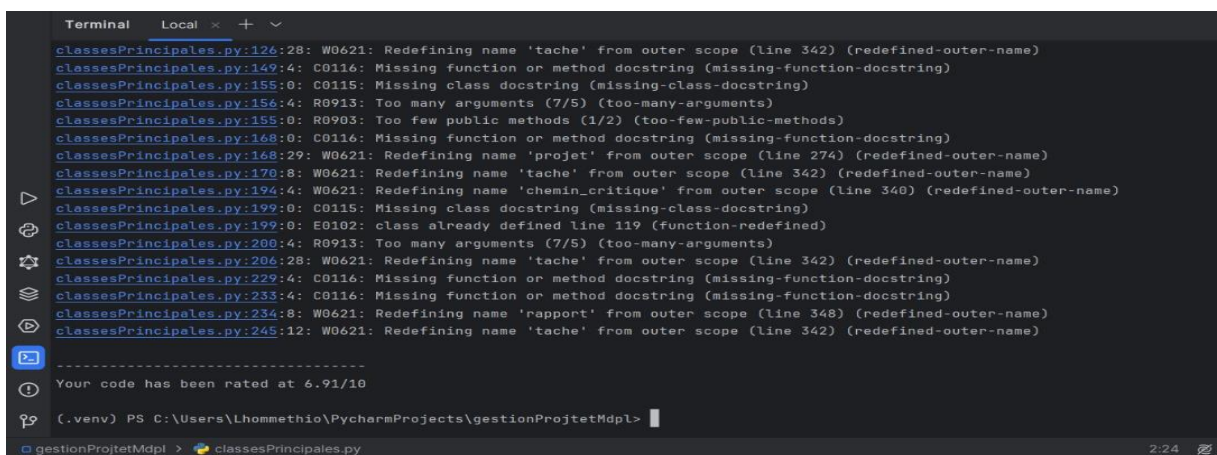
Nous avons effectué plusieurs tentatives jusqu'à pouvoir obtenir une version optimale.

- **pylint**

Suite à l'exécution de cette commande, voici ce que l'on note :



```
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> pylint C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classesPrincipales.py
***** Module classesPrincipales
classesPrincipales.py:1:0: C0114: Missing module docstring (missing-module-docstring)
classesPrincipales.py:1:0: C0103: Module name "classesPrincipales" doesn't conform to snake_case naming style (invalid-name)
classesPrincipales.py:6:0: C0115: Missing class docstring (missing-class-docstring)
classesPrincipales.py:6:0: R0903: Too few public methods (0/2) (too-few-public-methods)
classesPrincipales.py:13:0: C0115: Missing class docstring (missing-class-docstring)
classesPrincipales.py:17:4: C0116: Missing function or method docstring (missing-function-docstring)
classesPrincipales.py:13:0: R0903: Too few public methods (1/2) (too-few-public-methods)
classesPrincipales.py:22:0: C0115: Missing class docstring (missing-class-docstring)
classesPrincipales.py:22:0: R0903: Too few public methods (0/2) (too-few-public-methods)
classesPrincipales.py:29:0: C0115: Missing class docstring (missing-class-docstring)
classesPrincipales.py:29:0: R0903: Too few public methods (0/2) (too-few-public-methods)
classesPrincipales.py:37:0: C0115: Missing class docstring (missing-class-docstring)
classesPrincipales.py:38:4: R0913: Too many arguments (7/5) (too-many-arguments)
classesPrincipales.py:48:4: C0116: Missing function or method docstring (missing-function-docstring)
classesPrincipales.py:48:33: W0621: Redefining name 'tache' from outer scope (line 342) (redefined-outer-name)
classesPrincipales.py:37:0: R0903: Too few public methods (1/2) (too-few-public-methods)
classesPrincipales.py:53:0: C0115: Missing class docstring (missing-class-docstring)
classesPrincipales.py:53:0: R0903: Too few public methods (0/2) (too-few-public-methods)
classesPrincipales.py:61:0: C0115: Missing class docstring (missing-class-docstring)
classesPrincipales.py:61:0: R0902: Too many instance attributes (10/7) (too-many-instance-attributes)
```



```
classesPrincipales.py:126:28: W0621: Redefining name 'tache' from outer scope (line 342) (redefined-outer-name)
classesPrincipales.py:149:4: C0116: Missing function or method docstring (missing-function-docstring)
classesPrincipales.py:155:0: C0115: Missing class docstring (missing-class-docstring)
classesPrincipales.py:156:4: R0913: Too many arguments (7/5) (too-many-arguments)
classesPrincipales.py:155:0: R0903: Too few public methods (1/2) (too-few-public-methods)
classesPrincipales.py:168:0: C0116: Missing function or method docstring (missing-function-docstring)
classesPrincipales.py:168:29: W0621: Redefining name 'projet' from outer scope (line 274) (redefined-outer-name)
classesPrincipales.py:170:8: W0621: Redefining name 'tache' from outer scope (line 342) (redefined-outer-name)
classesPrincipales.py:194:4: W0621: Redefining name 'chemin_critique' from outer scope (line 340) (redefined-outer-name)
classesPrincipales.py:199:0: C0115: Missing class docstring (missing-class-docstring)
classesPrincipales.py:199:0: E0102: class already defined line 119 (function-redefined)
classesPrincipales.py:206:4: R0913: Too many arguments (7/5) (too-many-arguments)
classesPrincipales.py:206:28: W0621: Redefining name 'tache' from outer scope (line 342) (redefined-outer-name)
classesPrincipales.py:229:4: C0116: Missing function or method docstring (missing-function-docstring)
classesPrincipales.py:233:4: C0116: Missing function or method docstring (missing-function-docstring)
classesPrincipales.py:234:8: W0621: Redefining name 'rapport' from outer scope (line 348) (redefined-outer-name)
classesPrincipales.py:245:12: W0621: Redefining name 'tache' from outer scope (line 342) (redefined-outer-name)
-----
Your code has been rated at 6.91/10
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp>
```

Voici les suggestions présentées ainsi qu'une note globale de notre code présentant une note assez bien mais améliorables.



A travers cela, nous avons effectué des modifications possibles pour améliorer la qualité de notre code en suivant les instructions :

```
Terminal Local x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> pylint C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classes_principales.py
***** Module classes_principales
classes_principales.py:1:0: C0114: Missing module docstring (missing-module-docstring)
...
classes_principales.py:5:0: R0903: Too few public methods (0/2) (too-few-public-methods)
classes_principales.py:12:0: R0903: Too few public methods (1/2) (too-few-public-methods)
classes_principales.py:22:0: R0903: Too few public methods (0/2) (too-few-public-methods)
classes_principales.py:29:0: R0903: Too few public methods (0/2) (too-few-public-methods)
classes_principales.py:39:4: R0913: Too many arguments (7/5) (too-many-arguments)
classes_principales.py:49:33: W0621: Redefining name 'tache' from outer scope (line 316) (redefined-outer-name)
classes_principales.py:37:0: R0903: Too few public methods (1/2) (too-few-public-methods)
classes_principales.py:54:0: R0903: Too few public methods (0/2) (too-few-public-methods)
classes_principales.py:62:0: R0902: Too many instance attributes (10/7) (too-many-instance-attributes)
classes_principales.py:64:4: R0913: Too many arguments (6/5) (too-many-arguments)
classes_principales.py:77:28: W0621: Redefining name 'tache' from outer scope (line 316) (redefined-outer-name)
classes_principales.py:100:4: C0116: Missing function or method docstring (missing-function-docstring)
classes_principales.py:98:0: R0903: Too few public methods (1/2) (too-few-public-methods)
classes_principales.py:104:0: R0903: Too few public methods (1/2) (too-few-public-methods)
classes_principales.py:110:0: R0903: Too few public methods (1/2) (too-few-public-methods)
classes_principales.py:116:0: R0903: Too few public methods (1/2) (too-few-public-methods)
classes_principales.py:128:4: R0913: Too many arguments (7/5) (too-many-arguments)
classes_principales.py:134:28: W0621: Redefining name 'tache' from outer scope (line 316) (redefined-outer-name)
classes_principales.py:167:8: W0621: Redefining name 'rapport' from outer scope (line 322) (redefined-outer-name)
classes_principales.py:178:12: W0621: Redefining name 'tache' from outer scope (line 316) (redefined-outer-name)
classes_principales.py:204:4: R0913: Too many arguments (7/5) (too-many-arguments)
classes_principales.py:202:0: R0903: Too few public methods (1/2) (too-few-public-methods)
classes_principales.py:215:29: W0621: Redefining name 'projet' from outer scope (line 248) (redefined-outer-name)
classes_principales.py:218:8: W0621: Redefining name 'tache' from outer scope (line 316) (redefined-outer-name)
classes_principales.py:241:4: W0621: Redefining name 'chemin_critique' from outer scope (line 314) (redefined-outer-name)
-----
Your code has been rated at 8.49/10 (previous run: 8.49/10, +0.00)
```

Le constat est qu'on a pu avoir une évolution considérable de notre globale qui est une très bonne note.

- **mypy**

La commande *mypy* nous montre ici les effets de son exécution :

```
Terminal Local x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> mypy C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classes_Principales.py
classes_Principales.py:17: note: By default the bodies of untyped functions are not checked, consider using --check-untyped-defs [annotation-unchecked]
classes_Principales.py:273: error: "Tache" has no attribute "early_start" [attr-defined]
classes_Principales.py:274: error: "Tache" has no attribute "early_finish" [attr-defined]
classes_Principales.py:274: error: "Tache" has no attribute "early_start" [attr-defined]
classes_Principales.py:274: error: "Tache" has no attribute "duree" [attr-defined]
classes_Principales.py:284: error: "Tache" has no attribute "late_start" [attr-defined]
Found 5 errors in 1 file (checked 1 source file)
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp>
```

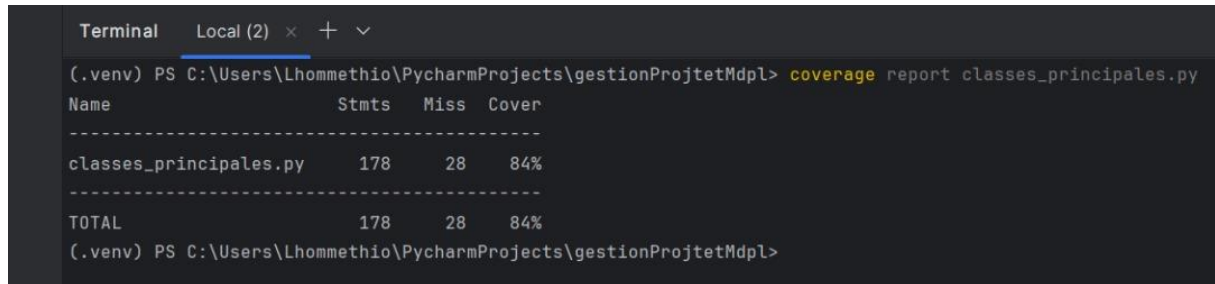
Nous pouvons noter que l'exécution nous renvoie des erreurs qui se réfèrent aux attributs de la classe *Tache*.

```
Terminal Local x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> mypy C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classes_Principales.py
classes_Principales.py:17: note: By default the bodies of untyped functions are not checked, consider using --check-untyped-defs [annotation-unchecked]
Success: no issues found in 1 source file
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp>
```

Nous avons effectué des modifications pour la rendre plus optimal.

- **coverage**

L'exécution de la commande *coverage* nous renvoie l'analyse de la couverture de code de nos tests précédents et voilà ce que l'on obtient :

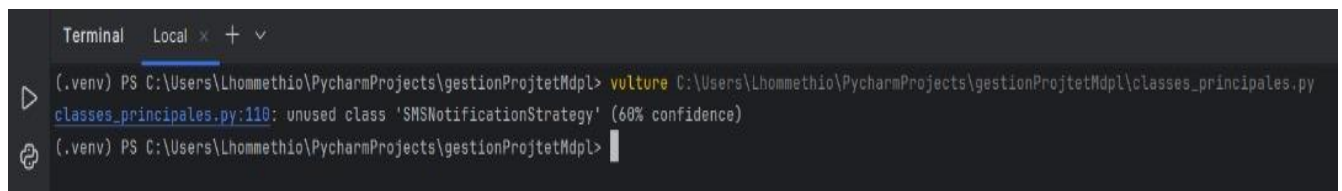


```
Terminal Local (2) x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl> coverage report classes_principales.py
Name                               Stmts  Miss  Cover
-----
classes_principales.py             178    28    84%
TOTAL                             178    28    84%
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl>
```

En analysant ces résultats, on peut souligner que l'on a **178** lignes de code renvoyés par *Stmts*, **28** par *Miss* qui veut dire que **28** lignes de code n'ont pas été exécutés par les tests et **84%** de *Cover* qui prouvent que **84%** des instructions ont été couvertes par les tests.

- **vulture**

L'utilisation de la commande *vulture* nous permet de constater ceci :

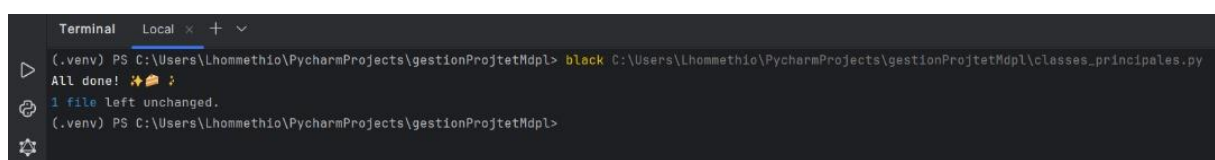


```
Terminal Local x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl> vulture C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classes_principales.py
classes_principales.py:110: unused class 'SMSNotificationsStrategy' (60% confidence)
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl>
```

Nous pouvons faire la remarque que *SMSNotificationsStrategy* est une classe déclarée qui n'a pas été utilisée. Ceci peut être justifié par la non utilisation de la classe lors de l'exécution dans la main et comme dans l'exemple du Log A, la notification par SMS n'y est pas d'où ce choix.

- **Black**

Après avoir utilisé la commande *black*, voici ce que l'on peut observer :



```
Terminal Local x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl> black C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl\classes_principales.py
All done! 🎉👏👏
1 file left unchanged.
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdpl>
```

On peut noter que le fichier n'a pas été modifié d'où l'on peut retenir que le fichier est déjà conforme à l'utilisation de *Black*.

- **radon**

La commande *radon* après application nous renvoie ces données ci-dessous :

```

(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\> radon cc C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classes_principales.py
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classes_principales.py
...
  F 264:0 calculer_chemin_critique - C
  M 198:4 ProjetNotifiable.generer_rapport_activite - B
  C 5:0 Membre - A
  C 13:0 Equipe - A
  C 24:0 Jalon - A
  C 32:0 Risque - A
  C 41:0 Tache - A
  C 66:0 Changement - A
  C 75:0 Projet - A
  C 118:0 NotificationStrategy - A
  C 125:0 EmailNotificationStrategy - A
  C 132:0 SMSNotificationStrategy - A
  C 139:0 NotificationContext - A
  C 150:0 ProjetNotifiable - A
  M 193:4 ProjetNotifiable.notifier_tous_membres - A
  C 243:0 TacheExt - A
  M 8:4 Membre.__init__ - A
  M 16:4 Equipe.__init__ - A
  M 19:4 Equipe.ajouter_membre - A
  M 27:4 Jalon.__init__ - A
  M 35:4 Risque.__init__ - A
  M 44:4 Tache.__init__ - A
  M 61:4 Tache.ajouter_dependance - A
  M 69:4 Changement.__init__ - A
  M 78:4 Projet.__init__ - A
  M 97:4 Projet.ajouter_tache - A
  M 101:4 Projet.ajouter_membre_equipe - A
  M 105:4 Projet.ajouter_risque - A
  M 109:4 Projet.ajouter_jalon - A
  M 113:4 Projet.enregistrer_changement - A
  M 113:4 Projet.enregistrer_changement - A
  M 121:4 NotificationStrategy.envoyer - A
  M 128:4 EmailNotificationStrategy.envoyer - A
  M 135:4 SMSNotificationStrategy.envoyer - A
  M 142:4 NotificationContext.__init__ - A
  M 145:4 NotificationContext.notifier - A
  M 153:4 ProjetNotifiable.__init__ - A
  M 165:4 ProjetNotifiable.ajouter_tache - A
  M 170:4 ProjetNotifiable.ajouter_membre_equipe - A
  M 175:4 ProjetNotifiable.ajouter_risque - A
  M 180:4 ProjetNotifiable.ajouter_jalon - A
  M 185:4 ProjetNotifiable.enregistrer_changement - A
  M 246:4 TacheExt.__init__ - A
  (.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\>
  
```

La remarque qui frappe est que pour chaque classe on obtient des lettres qui reflètent la complexité cyclomatique et la structuration globale. On obtient **A** partout signifiant que le risque faible et un block simple. Ce qui rapproche notre code du coté optimal.

- **pyflakes**

Nous avons exécuté pyflakes et à travers l'illustration on peut voir le résultat :

```

Terminal Local x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\> pyflakes C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classes_principales.py
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\> pyflakes C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classes_principales.py
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\> pyflakes C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\classes_principales.py
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\>
  
```

Il n'y a aucune note qui apparait donc il n'y a pas d'erreurs de syntaxes ou style dans le code.

## 2. Phase\_test.py :

Nous allons tenter d'utiliser les commandes suivantes dans notre fichier de test :

- **flake8**

Avec cette commande, on obtient les faits suivants :

```
Terminal Local (2) x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\> flake8 C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py:4:80: E501 line too long (96 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py:13:80: E501 line too long (85 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py:74:80: E501 line too long (86 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py:78:80: E501 line too long (109 > 79 characters)
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py:86:80: E501 line too long (101 > 79 characters)
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\>
```

Comme dans l'autre fichier, on constate que plusieurs des lignes signalent que les nombres de caractères dépassent la règle E501 qui stipule que les lignes de code ne doivent pas dépasser 79 caractères.

- **pylint**

Les résultats obtenus après l'utilisation de la commande *pylint* se présentent comme suit :

```
Terminal Local (2) x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\> pylint C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
***** Module phase_test
phase_test.py:78:0: C0301: Line too long (109/100) (line-too-long)
phase_test.py:86:0: C0301: Line too long (101/100) (line-too-long)
phase_test.py:1:0: C0114: Missing module docstring (missing-module-docstring)
phase_test.py:8:0: C0115: Missing class docstring (missing-class-docstring)
phase_test.py:8:0: R0902: Too many instance attributes (8/7) (too-many-instance-attributes)
phase_test.py:66:4: C0116: Missing function or method docstring (missing-function-docstring)
phase_test.py:71:4: C0116: Missing function or method docstring (missing-function-docstring)
phase_test.py:76:4: C0116: Missing function or method docstring (missing-function-docstring)
phase_test.py:80:4: C0116: Missing function or method docstring (missing-function-docstring)
phase_test.py:84:4: C0116: Missing function or method docstring (missing-function-docstring)
phase_test.py:88:4: C0116: Missing function or method docstring (missing-function-docstring)
phase_test.py:95:4: C0116: Missing function or method docstring (missing-function-docstring)
-----
Your code has been rated at 7.78/10
```

Voici la note ainsi que les remarques à faire dans notre code. Nous avons par la suite tenté d'améliorer et voici le résultat :

```
Terminal Local (2) x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\> pylint C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
***** Module phase_test
phase_test.py:1:0: C0114: Missing module docstring (missing-module-docstring)
phase_test.py:8:0: C0115: Missing class docstring (missing-class-docstring)
phase_test.py:8:0: R0902: Too many instance attributes (8/7) (too-many-instance-attributes)
phase_test.py:66:4: C0116: Missing function or method docstring (missing-function-docstring)
phase_test.py:71:4: C0116: Missing function or method docstring (missing-function-docstring)
phase_test.py:76:4: C0116: Missing function or method docstring (missing-function-docstring)
phase_test.py:81:4: C0116: Missing function or method docstring (missing-function-docstring)
phase_test.py:85:4: C0116: Missing function or method docstring (missing-function-docstring)
phase_test.py:90:4: C0116: Missing function or method docstring (missing-function-docstring)
phase_test.py:97:4: C0116: Missing function or method docstring (missing-function-docstring)
-----
Your code has been rated at 8.15/10 (previous run: 7.78/10, +0.37)
```

Le résultat stipule que les remarques ont considérablement diminués ainsi que le note qui a bien évolué.



- mypy

Comme on peut le constater, voici ce que l'on obtient avec *mypy* :

```
Terminal Local (2) x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> mypy C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
classes_principales.py:17: note: By default the bodies of untyped functions are not checked, consider using --check-untyped-defs [annotation-unchecked]
Success: no issues found in 1 source file
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> |
```

Nous avons aperçu comme résultat que le code semble optimal qu'il n'y a aucune issue modifiable trouvée. Ce qui est très rassurant.

- coverage

L'application de cette commande nous montre des résultats statistiques interprétables. Voici ce que l'on obtient à l'affichage :

```
Terminal Local (2) x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> coverage report phase_test.py
Name           Stmts  Miss  Cover
-----
phase_test.py   54      1   98%
-----
TOTAL           54      1   98%
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp>
```

En interprétant ceci, nous pouvons observer qu'il y'a **54** lignes de code dans *Stmts* avec **1** dans *Miss* qui veut dire qu'une seule ligne de code n'a pu être exécuté par les tests et pour finir **98%** de *Cover* montrant que **98%** des instructions ont été effectuées par les tests.

- vulture

Nous avons tenté d'utiliser la commande d'analyse vulture et après exécution, voici ce que nous notons :

```
Terminal Local (2) x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> vulture C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> vulture C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> vulture C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp> |
```

On peut observer qu'il n'y a aucun résultat alors cette commande est censée détecter des variables inutilisées d'où l'on peut retenir que le code ne contient pas de variables inutilisées.

- **Black**

La commande utilisée sur le fichier nous montre les faits suivants :

```
Terminal Local (2) x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\> black C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
reformatted C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py

All done! ✨ 🍰 ✨
1 file reformatted.
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\>
```

On peut faire un constat qui avec lequel le fichier a été modifié d'où l'on pourrait en déduire que le code est déjà en normes à l'utilisation de Black.

- **radon**

Les résultats obtenus à l'utilisation de la commande radon nous montre les effets suivants :

```
Terminal Local (2) x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\> radon cc C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
C 16:0 TestProjet - A
M 98:4 TestProjet.test_chemin_critique - A
M 18:4 TestProjet.setUp - A
M 71:4 TestProjet.test_ajouter_membre - A
M 76:4 TestProjet.test_ajouter_tache - A
M 81:4 TestProjet.test_ajouter_risque - A
M 88:4 TestProjet.test_ajouter_jalon - A
M 92:4 TestProjet.test_enregistrer_changement - A
M 105:4 TestProjet.test_generer_rapport_activite - A
```

Nous pouvons constater que pour toutes les classes que l'on a des **A** partout signifiant que le risque faible et un block simple. D'où l'on pourrait être confiant que notre code se rapproche de l'optimal.

- **pyflakes**

Après l'utilisation de test de la commande, voici ce que nous remarquons :

```
Terminal Local (2) x + v
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\> pyflakes C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\> pyflakes C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\> pyflakes C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\phase_test.py
(.venv) PS C:\Users\Lhommethio\PycharmProjects\gestionProjetMdp\>
```

La remarque générale faite c'est qu'il n'y a aucune note visible d'où l'on peut retenir qu'il n'y a pas d'erreurs de syntaxes ou style dans le code de notre fichier.

En Résumé, nous pouvons dire que les outils d'analyse de code nous ont non seulement permis de mieux organiser la structure(forme) de notre code, mais de d'optimiser beaucoup plus le fond de notre code en améliorant la qualité de celle-ci afin de la rendre bien plus simple, concis et clair à comprendre.

Les outils d'outils d'analyse de code ont été d'une importance considérable pour justifier la mesure et la qualité de notre code.