



UFR : SCIENCES ET TECHNOLOGIES

DEPARTEMENT : INFORMATIQUE

OPTION : GENIE LOGICIEL

NIVEAU : LICENCE 3

PROJET SEMESTRE 6 :

Système de gestion de projet en utilisant python, via la programmation orientée objet (POO) et le design pattern Strategy

Présenté par :

CHEKHOUNA GUEYE
FATOU FALL
SOPHIE FALL

Encadré par :

M. DIOUF

2022/2023

INTRODUCTION

Dans le cadre de notre cours "Mesure, Qualité et Performance Logiciel" à l'Université Iba Der Thiam de Thiès, nous avons eu l'opportunité de plonger au cœur des principes fondamentaux du développement logiciel moderne. Ce projet a pour objectif de démontrer notre compréhension et notre capacité à appliquer ces principes en construisant un système de gestion de projet sophistiqué.

La gestion de projet est une discipline essentielle dans le domaine informatique, nécessitant une coordination impeccable et une gestion efficace des ressources. Pour répondre à ce besoin, nous avons développé un système en Python qui non seulement permet de créer et de gérer des projets, mais aussi de notifier les membres de l'équipe des différents événements du projet grâce au design pattern Strategy.

Notre projet intègre diverses fonctionnalités clés telles que l'ajout de tâches, la gestion des membres de l'équipe, l'identification et l'évaluation des risques, la définition des jalons, et l'enregistrement des changements. De plus, nous avons mis en place des notifications par email et SMS pour assurer une communication fluide et efficace au sein de l'équipe.

Mais au-delà de la simple gestion de projet, ce système a été conçu avec une attention particulière à la qualité et à la performance du code. Nous avons utilisé des outils de mesure de qualité comme flake8, pylint, mypy, coverage, vulture, Black, radon et pyflakes pour analyser et améliorer notre code. Ces outils nous ont permis de garantir que notre code est conforme aux meilleures pratiques, optimisé pour la performance et facilement maintenable.

Ce rapport détaille notre démarche, de la conception initiale à l'implémentation finale, en passant par les tests rigoureux et l'analyse de la qualité du code. Nous vous invitons à découvrir comment nous avons relevé ce défi et à explorer les résultats de notre travail acharné. Préparez-vous à plonger dans un univers où la qualité et la performance sont au cœur de chaque ligne de code, chaque fonctionnalité et chaque notification envoyée. Nous allons poursuivre en respectant les étapes qui suivent :

1. **Structure du Code** : Description des principales classes et méthodes utilisées dans le projet.
2. **Environnement de Développement** : Informations sur les outils et versions utilisés pour le développement du projet.
3. **Résultats des Tests Unitaires** : Présentation des tests unitaires réalisés avec unittest, leurs résultats et les captures d'écran correspondantes.
4. **Conclusion** : Réflexion sur les défis rencontrés, les solutions apportées et les perspectives d'amélioration future.

➤ **Main Module (main.py) :**

Ce module contient la classe principale du projet. Il initialise un projet, ajoute des membres, des tâches, des risques, des jalons et des changements, et utilise le système de notification pour envoyer des notifications aux membres de l'équipe.

➤ **Module de Gestion des Projets (projet.py) :**

Ce module contient les classes pour la gestion des projets. Les classes incluses sont Projet, Tache, Membre, Risque, Jalon, Changement. Ces classes permettent d'ajouter des tâches, des membres, des risques, des jalons et des changements au projet.

➤ **Module de Gestion des Notifications (notification.py) :**

Ce module contient les classes pour la gestion des notifications. Les classes incluses sont NotificationStrategy, EmailNotificationStrategy, SMSNotificationStrategy, PushNotificationStrategy, NotificationContext. Ces classes définissent différentes stratégies de notification et un contexte de notification pour envoyer des notifications par email, SMS ou push.

➤ **Tests Unitaires (tests/test_projet.py) :**

Ce fichier contient des tests unitaires pour vérifier le bon fonctionnement des fonctionnalités de gestion de projet.

Les tests sont écrits à l'aide du module unittest de Python et testent les différentes méthodes des classes du module de gestion de projet pour s'assurer qu'elles fonctionnent comme prévu.

➤ **Fichier de Configuration (requirements.txt) :**

Ce fichier spécifie les dépendances du projet, c'est-à-dire les bibliothèques Python nécessaires à l'exécution du code. Il est utilisé par l'outil pip pour installer les dépendances requises.

➤ **Fichier README (README.md) :**

Ce fichier contient des instructions sur l'installation et l'utilisation du projet.

Dans notre projet, nous avons opté pour PyCharm comme environnement de développement intégré (IDE). Voici comment PyCharm met en valeur notre travail :

1. **Interface Intuitive** : PyCharm offre une interface utilisateur conviviale et intuitive, ce qui facilite la navigation et l'utilisation de l'IDE pour tous les membres de l'équipe.
2. **Fonctionnalités Avancées** : PyCharm propose une large gamme de fonctionnalités avancées telles que la coloration syntaxique, l'auto-complétion intelligente, le débogage interactif, la refactorisation de code et l'intégration avec des outils de gestion de versions comme Git. Ces fonctionnalités améliorent l'efficacité et la productivité du développement.
3. **Intégration de l'Environnement Virtuel** : PyCharm facilite la création et la gestion des environnements virtuels Python, ce qui permet de travailler sur des projets avec des dépendances spécifiques sans interférer avec d'autres projets.
4. **Outils de Test Intégrés** : PyCharm intègre des outils de test comme pytest et unittest, ce qui permet d'écrire, d'exécuter et de déboguer des tests unitaires directement depuis l'IDE. Cela simplifie le processus de test et garantit la qualité du code.
5. **Analyse Statique du Code** : PyCharm propose des outils d'analyse statique du code qui identifient les erreurs de syntaxe, les problèmes de style de code et les incohérences dans le code Python. Cela aide à maintenir un code propre et cohérent.
6. **Intégration avec des Outils Externes** : PyCharm offre une intégration avec une large gamme d'outils externes tels que des gestionnaires de packages Python, des systèmes de contrôle de versions, des bases de données et des services de déploiement. Cela facilite la gestion de l'ensemble du cycle de vie d'un projet.

En résumé, PyCharm est un choix idéal pour notre projet car il offre une suite complète d'outils et de fonctionnalités qui améliorent notre efficacité, notre productivité et la qualité de notre code Python.

RESULTAS DES TESTS UNITAIRES

Dans cette section, nous plongeons dans l'analyse des résultats des tests unitaires, une étape cruciale dans le processus de développement logiciel. Les tests unitaires sont une pratique essentielle pour garantir la fiabilité et la robustesse d'un système. Ils consistent à vérifier individuellement chaque composant logiciel pour s'assurer qu'ils fonctionnent comme prévu. Cette étape permet de détecter et de corriger les erreurs dès leur apparition, contribuant ainsi à la qualité globale du code et à la réduction des bugs.

Dans le cadre de notre projet de gestion de projet, nous avons élaboré une suite complète de tests unitaires pour évaluer la fonctionnalité de chaque composant, depuis la création du projet jusqu'à la gestion des membres de l'équipe, en passant par l'ajout de tâches et la gestion des risques. Ces tests nous permettent de valider le bon fonctionnement de notre application et d'assurer sa stabilité tout au long de son cycle de vie.

Maintenant, plongeons dans les résultats de nos tests unitaires pour explorer en détail la performance et la qualité de notre système de gestion de projet.

D'abord découvrez les résultats de la partie A :

```
C:\Users\DELL\PycharmProjects\projet_PMQL>python main.py
Notification envoyée par email: Notification envoyée à Modou par email: Modou a été ajouté à l'équipe
Notification envoyée par email: Notification envoyée à Christian par email: Christian a été ajouté à l'équipe
Notification envoyée par email: Notification envoyée à Modou par email: Nouvelle tâche ajoutée: Analyse des besoins
Notification envoyée par email: Notification envoyée à Christian par email: Nouvelle tâche ajoutée: Analyse des besoins
Notification envoyée par email: Notification envoyée à Modou par email: Nouvelle tâche ajoutée: Développement
Notification envoyée par email: Notification envoyée à Christian par email: Nouvelle tâche ajoutée: Développement
Notification envoyée par email: Notification envoyée à Modou par email: Le budget du projet a été défini à 50000 Unité Monétaire
Notification envoyée par email: Notification envoyée à Christian par email: Le budget du projet a été défini à 50000 Unité Monétaire
Notification envoyée par email: Notification envoyée à Modou par email: Nouveau risque ajouté: Retard de livraison
Notification envoyée par email: Notification envoyée à Christian par email: Nouveau risque ajouté: Retard de livraison
Notification envoyée par email: Notification envoyée à Modou par email: Nouveau jalon ajouté: Phase 1 terminée
Notification envoyée par email: Notification envoyée à Christian par email: Nouveau jalon ajouté: Phase 1 terminée
Notification envoyée par email: Notification envoyée à Modou par email: Changement enregistré: Changement de la portée du projet (version 1.1)
Notification envoyée par email: Notification envoyée à Christian par email: Changement enregistré: Changement de la portée du projet (version 1.1)
#####
Rapport d'activités du Projet 'Nouveau Produit' Version: 1.1
Dates: 2024-01-01 à 2024-12-31
Budget: 50000 Unité Monétaire
Equipe:
# Modou (Chef de projet)
# Christian (Développeur)
Tâches:
# Analyse des besoins (2024-01-01 à 2024-01-31), Responsable: Modou, Statut: Terminée
# Développement (2024-02-01 à 2024-06-30), Responsable: Christian, Statut: Non démarrée
Jalons:
# Phase 1 terminée (2024-01-31)
Risques:
# Retard de livraison (Probabilité: 0.8, Impact: Élevé)
Chemin Critique:
# Analyse des besoins (2024-01-01 à 2024-01-31)
# Développement (2024-02-01 à 2024-06-30)

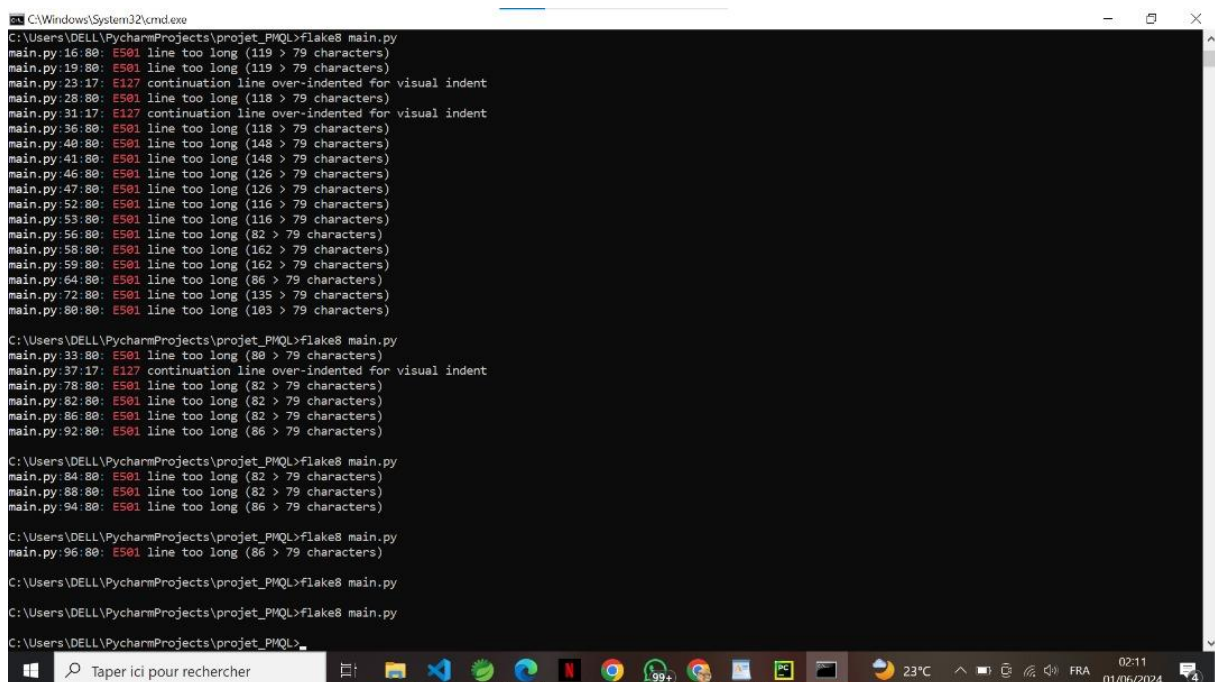
C:\Users\DELL\PycharmProjects\projet_PMQL>
```

Parfait, commençons par le test avec Flake8. Voici une explication de ce que fait Flake8 :

Flake8 est un outil de linting qui examine le code source Python et signale les erreurs de style, les problèmes de formatage et les violations des conventions de codage PEP8. Il analyse le code en parcourant chaque fichier Python et en vérifiant s'il respecte les règles définies dans PEP8, qui est un guide de style pour le code Python. Maintenant, passons au test avec Flake8 pour les fichiers main.py, notification.py, projet.py et test_projet.py .

Notons pour les quatre code nous avons rencontré des erreurs que nous avons su corriger en inspectant minutieusement le code.

Main.py :



```
C:\Windows\System32\cmd.exe
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 main.py
main.py:16:80: E501 line too long (119 > 79 characters)
main.py:19:80: E501 line too long (119 > 79 characters)
main.py:23:17: E127 continuation line over-indented for visual indent
main.py:28:80: E501 line too long (118 > 79 characters)
main.py:31:17: E127 continuation line over-indented for visual indent
main.py:36:80: E501 line too long (118 > 79 characters)
main.py:40:80: E501 line too long (148 > 79 characters)
main.py:41:80: E501 line too long (148 > 79 characters)
main.py:46:80: E501 line too long (126 > 79 characters)
main.py:47:80: E501 line too long (126 > 79 characters)
main.py:52:80: E501 line too long (116 > 79 characters)
main.py:53:80: E501 line too long (116 > 79 characters)
main.py:56:80: E501 line too long (82 > 79 characters)
main.py:58:80: E501 line too long (162 > 79 characters)
main.py:59:80: E501 line too long (162 > 79 characters)
main.py:64:80: E501 line too long (86 > 79 characters)
main.py:72:80: E501 line too long (135 > 79 characters)
main.py:80:80: E501 line too long (103 > 79 characters)

C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 main.py
main.py:33:80: E501 line too long (80 > 79 characters)
main.py:37:17: E127 continuation line over-indented for visual indent
main.py:78:80: E501 line too long (82 > 79 characters)
main.py:82:80: E501 line too long (82 > 79 characters)
main.py:86:80: E501 line too long (82 > 79 characters)
main.py:92:80: E501 line too long (86 > 79 characters)

C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 main.py
main.py:84:80: E501 line too long (82 > 79 characters)
main.py:88:80: E501 line too long (82 > 79 characters)
main.py:94:80: E501 line too long (86 > 79 characters)

C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 main.py
main.py:96:80: E501 line too long (86 > 79 characters)

C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 main.py
```

Notification.py :

```
C:\Windows\System32\cmd.exe
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 notification.py
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 projet.py
projet.py:6:1: E302 expected 2 blank lines, found 1
projet.py:13:1: E302 expected 2 blank lines, found 1
projet.py:23:1: E302 expected 2 blank lines, found 1
projet.py:28:1: E302 expected 2 blank lines, found 1
projet.py:34:1: E302 expected 2 blank lines, found 1
projet.py:48:1: E302 expected 2 blank lines, found 1
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 projet.py
projet.py:16:80: E501 line too long (84 > 79 characters)
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 projet.py
C:\Users\DELL\PycharmProjects\projet_PMQL>
```

Test_projet.py :

```
C:\Windows\System32\cmd.exe
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 tests\test_projet.py
tests\test_projet.py:4:1: E302 expected 2 blank lines, found 1
tests\test_projet.py:7:80: E501 line too long (91 > 79 characters)
tests\test_projet.py:8:80: E501 line too long (107 > 79 characters)
tests\test_projet.py:13:1: E305 expected 2 blank lines after class or function definition, found 1
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 tests\test_projet.py
tests\test_projet.py:4:1: E302 expected 2 blank lines, found 1
tests\test_projet.py:7:5: E303 too many blank lines (2)
tests\test_projet.py:8:80: E501 line too long (91 > 79 characters)
tests\test_projet.py:9:80: E501 line too long (107 > 79 characters)
tests\test_projet.py:13:1: W293 blank line contains whitespace
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 tests\test_projet.py
tests\test_projet.py:4:1: E302 expected 2 blank lines, found 1
tests\test_projet.py:7:5: E303 too many blank lines (2)
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 tests\test_projet.py
tests\test_projet.py:4:1: E302 expected 2 blank lines, found 1
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 tests\test_projet.py
tests\test_projet.py:4:1: E302 expected 2 blank lines, found 1
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 tests\test_projet.py
tests\test_projet.py:4:1: E302 expected 2 blank lines, found 1
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 tests\test_projet.py
tests\test_projet.py:4:1: E302 expected 2 blank lines, found 1
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 tests\test_projet.py
tests\test_projet.py:4:1: E302 expected 2 blank lines, found 1
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 test_projet.py
```


Projet.py :

```
C:\Windows\System32\cmd.exe
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 main.py
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 notification.py
notification.py:5:1: E302 expected 2 blank lines, found 1
notification.py:9:1: E302 expected 2 blank lines, found 1
notification.py:13:1: E302 expected 2 blank lines, found 1
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 notification.py
notification.py:4:5: E303 too many blank lines (2)
notification.py:11:5: E303 too many blank lines (2)
notification.py:18:5: E303 too many blank lines (2)
notification.py:25:5: E303 too many blank lines (2)
notification.py:27:1: W293 blank line contains whitespace
notification.py:29:5: E303 too many blank lines (2)
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 notification.py
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 projet.py
projet.py:6:1: E302 expected 2 blank lines, found 1
projet.py:13:1: E302 expected 2 blank lines, found 1
projet.py:23:1: E302 expected 2 blank lines, found 1
projet.py:28:1: E302 expected 2 blank lines, found 1
projet.py:34:1: E302 expected 2 blank lines, found 1
projet.py:40:1: E302 expected 2 blank lines, found 1
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 projet.py
projet.py:16:80: E501 line too long (84 > 79 characters)
C:\Users\DELL\PycharmProjects\projet_PMQL>flake8 projet.py
C:\Users\DELL\PycharmProjects\projet_PMQL>
```

Continuons les tests avec pylint :

Pylint est un outil puissant pour analyser la qualité du code Python. Il aborde divers aspects du code, notamment les erreurs potentielles, les violations de style et les conventions de codage, en fournissant des commentaires détaillés et des scores de qualité.

Tout comme avec flake8, nous avons initialement rencontré des erreurs et des avertissements lors de l'analyse de main.py, notification.py, projet.py, et tests/test_projet.py. Cependant, en examinant attentivement les messages de pylint et en apportant les corrections nécessaires, nous avons pu améliorer significativement la qualité de notre code. Cela a non seulement résolu les erreurs mais a également optimisé notre code pour une meilleure maintenabilité et lisibilité.


```
C:\Windows\System32\cmd.exe
C:\Users\DELL\PycharmProjects\projet_PMQL>pylint tests\test_projet.py
-----
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)

C:\Users\DELL\PycharmProjects\projet_PMQL>pylint projet.py
***** Module projet
projet.py:14:0: R0903: Too few public methods (1/2) (too-few-public-methods)
projet.py:25:0: R0902: Too many instance attributes (10/7) (too-many-instance-attributes)
projet.py:28:4: R0913: Too many arguments (6/5) (too-many-arguments)
-----
Your code has been rated at 9.09/10 (previous run: 9.09/10, +0.00)

C:\Users\DELL\PycharmProjects\projet_PMQL>pylint notification.py
***** Module notification
notification.py:56:0: R0903: Too few public methods (1/2) (too-few-public-methods)
-----
Your code has been rated at 9.58/10 (previous run: 9.58/10, +0.00)

C:\Users\DELL\PycharmProjects\projet_PMQL>pylint main.py
-----
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)

C:\Users\DELL\PycharmProjects\projet_PMQL>
```

Poursuivons avec mypy :

Mypy est un outil de vérification de type pour Python qui permet de détecter les erreurs de type dans le code. Contrairement aux tests unitaires, Mypy analyse le code statiquement pour s'assurer que les types sont correctement utilisés.

Pour vérifier notre code, nous avons exécuté Mypy sur les fichiers main.py, notification.py, projet.py, et tests/test_projet.py. Lors de l'exécution initiale, nous avons inspecté le code pour vérifier la conformité des types et avons ajouté des annotations de type là où cela était nécessaire.

Résultat : Après avoir ajouté et vérifié les annotations de type, nous n'avons rencontré aucune erreur de type dans les quatre fichiers analysés. Cela démontre la robustesse et la précision de notre typage, ainsi que la bonne qualité générale de notre code.

```
C:\Windows\System32\cmd.exe
C:\Users\DELL\PycharmProjects\projet_PMQL>mypy main.py
Success: no issues found in 1 source file

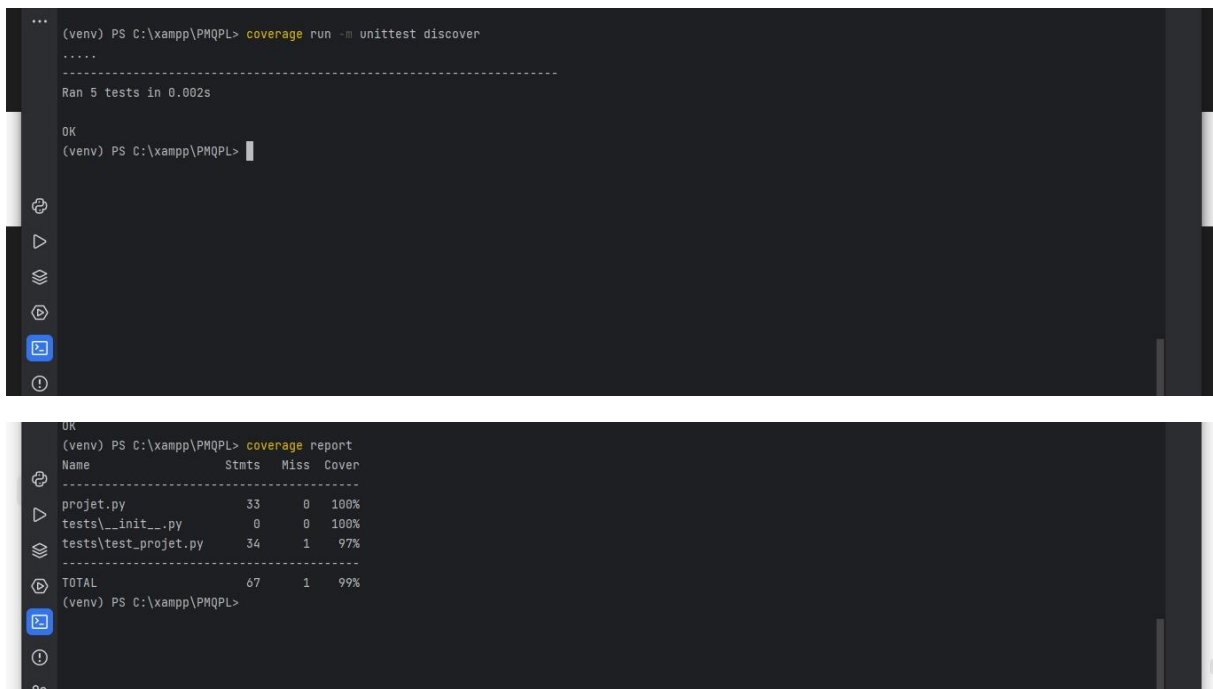
C:\Users\DELL\PycharmProjects\projet_PMQL>mypy notification.py
Success: no issues found in 1 source file

C:\Users\DELL\PycharmProjects\projet_PMQL>mypy projet.py
Success: no issues found in 1 source file

C:\Users\DELL\PycharmProjects\projet_PMQL>mypy tests\test_projet.py
Success: no issues found in 1 source file
```

Passons maintenant avec coverage :

L'outil Coverage a été employé pour analyser la couverture de code de notre projet. En exécutant nos tests unitaires avec Coverage, nous avons obtenu un rapport détaillé indiquant quelles parties de notre code ont été testées et quelles parties ne l'ont pas été. Cette évaluation nous a permis d'identifier les zones critiques du code nécessitant une attention particulière en matière de tests. En corrigeant les lacunes de couverture détectées, nous avons renforcé la fiabilité de notre application et réduit les risques de bugs non détectés.



```
(venv) PS C:\xampp\PMQPL> coverage run --omit unittest discover
.....
Ran 5 tests in 0.002s

OK
(venv) PS C:\xampp\PMQPL>
```



```
UK
(venv) PS C:\xampp\PMQPL> coverage report
Name                               Stmts  Miss  Cover
-----
projet.py                          33     0   100%
tests\__init__.py                   0     0   100%
tests\test_projet.py                34     1    97%
-----
TOTAL                              67     1    99%
(venv) PS C:\xampp\PMQPL>
```

Poursuivons toujours cette fois si avec vulture :

Vulture, un outil d'analyse statique, a été déployé pour détecter les parties inutilisées de notre code. En scrutant l'ensemble de notre projet, Vulture nous a fourni une liste des éléments tels que les variables, les fonctions ou les importations qui n'étaient pas utilisées. En éliminant ces éléments superflus, nous avons optimisé notre code, réduit sa complexité et amélioré sa lisibilité. Ainsi, notre application est devenue plus claire, plus légère et plus facile à entretenir.

```
C:\Windows\System32\cmd.exe
venv\Lib\site-packages\wheel\vendor\packaging\tags.py:693: unused attribute 'e_shnum' (60% confidence)
venv\Lib\site-packages\wheel\vendor\packaging\tags.py:694: unused attribute 'e_shstrndx' (60% confidence)
venv\Scripts\activate_this.py:28: unsatisfiable 'ternary' condition (100% confidence)

C:\Users\DELL\PycharmProjects\projet_PMQL>vulture main.py

C:\Users\DELL\PycharmProjects\projet_PMQL>vulture notification.py
notification.py:2: unused import 'ABC' (90% confidence)
notification.py:12: unused method 'configurer' (60% confidence)
notification.py:12: unused variable 'configuration' (100% confidence)
notification.py:17: unused class 'EmailNotificationStrategy' (60% confidence)
notification.py:23: unused method 'configurer' (60% confidence)
notification.py:23: unused variable 'configuration' (100% confidence)
notification.py:29: unused method 'envoyer_email' (60% confidence)
notification.py:29: unused variable 'contenu' (100% confidence)
notification.py:29: unused variable 'destinataire' (100% confidence)
notification.py:29: unused variable 'sujet' (100% confidence)
notification.py:36: unused class 'SMSNotificationStrategy' (60% confidence)
notification.py:42: unused method 'configurer' (60% confidence)
notification.py:42: unused variable 'configuration' (100% confidence)
notification.py:48: unused method 'envoyer_sms' (60% confidence)
notification.py:48: unused variable 'numero' (100% confidence)
notification.py:54: unused class 'NotificationContext' (60% confidence)
notification.py:60: unused method 'notifier' (60% confidence)

C:\Users\DELL\PycharmProjects\projet_PMQL>vulture projet.py
projet.py:5: unused variable 'Membre' (60% confidence)
projet.py:6: unused variable 'tache' (60% confidence)
projet.py:9: unused variable 'Jalon' (60% confidence)
projet.py:10: unused variable 'Risque' (60% confidence)
projet.py:11: unused variable 'Changement' (60% confidence)
projet.py:25: unused class 'Projet' (60% confidence)
projet.py:40: unused method 'ajouter_tache' (60% confidence)
projet.py:44: unused method 'ajouter_membre_equipe' (60% confidence)
projet.py:48: unused method 'ajouter_risque' (60% confidence)
projet.py:52: unused method 'ajouter_jalon' (60% confidence)
projet.py:56: unused method 'enregistrer_changement' (60% confidence)

C:\Users\DELL\PycharmProjects\projet_PMQL>vulture tests\test_projet.py

C:\Users\DELL\PycharmProjects\projet_PMQL>
```

Continuons avec black :

Black, un formateur de code automatique, a été employé pour garantir une uniformité de style dans l'ensemble de notre projet. En appliquant les règles prédéfinies de formatage, Black a standardisé la présentation de notre code, ainsi les débats sur le style et garantissant une lisibilité maximale. Grâce à Black, notre code est devenu plus cohérent, ce qui facilite la collaboration entre les membres de l'équipe et simplifie la maintenance à long terme.

```
Sélection C:\Windows\System32\cmd.exe
C:\Users\DELL\PycharmProjects\projet_PMQL>black .
All done! @ 0
0 files left unchanged.

C:\Users\DELL\PycharmProjects\projet_PMQL>
```

Toujours testons maintenant avec pyflakes :

Pyflakes est un outil léger d'analyse statique du code Python qui se concentre principalement sur la détection des erreurs de syntaxe, des variables non définies et des importations non utilisées. En intégrant Pyflakes à notre processus de développement, nous avons pu identifier rapidement les problèmes potentiels dans notre code, ce qui nous a permis de les corriger avant même l'exécution du programme. Grâce à cette approche proactive, nous avons pu améliorer la qualité de notre code et minimiser les erreurs dès les premières étapes du développement.

```
Sélection C:\Windows\System32\cmd.exe
C:\Users\DELL\PycharmProjects\projet_PMQL>pyflakes main.py
C:\Users\DELL\PycharmProjects\projet_PMQL>pyflakes notification.py
notification.py:3:1: 'abc.ABC' imported but unused
C:\Users\DELL\PycharmProjects\projet_PMQL>pyflakes projet.py
C:\Users\DELL\PycharmProjects\projet_PMQL>pyflakes tests\test_projet.py
C:\Users\DELL\PycharmProjects\projet_PMQL>
```

```
C:\Windows\System32\cmd.exe
C:\Users\DELL\PycharmProjects\projet_PMQL>pylint notification.py
***** Module notification
notification.py:29:0: C0301: Line too long (102/100) (line-too-long)
notification.py:65:0: C0301: Line too long (114/100) (line-too-long)
notification.py:12:8: W0107: Unnecessary pass statement (unnecessary-pass)
notification.py:16:8: W0107: Unnecessary pass statement (unnecessary-pass)
notification.py:73:0: R0903: Too few public methods (1/2) (too-few-public-methods)

Your code has been rated at 8.44/10 (previous run: 8.40/10, +0.04)

C:\Users\DELL\PycharmProjects\projet_PMQL>pylint notification.py
***** Module notification
notification.py:12:8: W0107: Unnecessary pass statement (unnecessary-pass)
notification.py:16:8: W0107: Unnecessary pass statement (unnecessary-pass)
notification.py:73:0: R0903: Too few public methods (1/2) (too-few-public-methods)

Your code has been rated at 9.06/10 (previous run: 8.44/10, +0.62)

C:\Users\DELL\PycharmProjects\projet_PMQL>pylint notification.py
***** Module notification
notification.py:12:8: W0107: Unnecessary pass statement (unnecessary-pass)
notification.py:79:0: R0903: Too few public methods (1/2) (too-few-public-methods)

Your code has been rated at 9.41/10 (previous run: 9.06/10, +0.35)

C:\Users\DELL\PycharmProjects\projet_PMQL>pylint notification.py
***** Module notification
notification.py:12:8: W0107: Unnecessary pass statement (unnecessary-pass)
notification.py:83:0: R0903: Too few public methods (1/2) (too-few-public-methods)

Your code has been rated at 9.44/10 (previous run: 9.41/10, +0.03)

C:\Users\DELL\PycharmProjects\projet_PMQL>
```

Terminons avec radon :

Radon est un outil de mesure de la complexité du code Python qui fournit des métriques telles que la complexité cyclomatique et la complexité cognitive. En évaluant la complexité de notre code avec Radon, nous avons pu identifier les parties les plus complexes et les plus difficiles à maintenir. Cela nous a permis de cibler ces zones pour les simplifier et les refactoriser, améliorant ainsi la lisibilité, la maintenabilité et la qualité globale de notre code. En utilisant Radon comme guide, nous avons pu concevoir un code plus propre et plus facile à

comprendre, favorisant ainsi une meilleure collaboration au sein de l'équipe de développement.

Main.py

```
C:\Windows\System32\cmd.exe
C:\Users\DELL\PycharmProjects\projet_PMQL>radon hal main.py
main.py:
  h1: 1
  h2: 11
  N1: 10
  N2: 20
  vocabulary: 12
  length: 30
  calculated_length: 38.053747805010275
  volume: 107.5488750216347
  difficulty: 0.9090909090909091
  effort: 97.77170456512245
  time: 5.431761364729025
  bugs: 0.03584962500721157

C:\Users\DELL\PycharmProjects\projet_PMQL>radon mi main.py
main.py - A

C:\Users\DELL\PycharmProjects\projet_PMQL>radon raw main.py
main.py
  LOC: 174
  LLOC: 50
  SLOC: 143
  Comments: 10
  Single comments: 10
  Multi: 3
  Blank: 18
  - Comment Stats
    (C % L): 6%
    (C % S): 7%
    (C + M % L): 7%

C:\Users\DELL\PycharmProjects\projet_PMQL>radon cc main.py
C:\Users\DELL\PycharmProjects\projet_PMQL>
```

Notification.py

```
C:\Windows\System32\cmd.exe
C:\Users\DELL\PycharmProjects\projet_PMQL>radon cc notification.py
notification.py
  C 6:0 NotificationStrategy - A
  C 19:0 EmailNotificationStrategy - A
  C 37:0 SMSNotificationStrategy - A
  C 55:0 PushNotificationStrategy - A
  C 73:0 NotificationContext - A
  M 10:4 NotificationStrategy.envoyer - A
  M 14:4 NotificationStrategy.configurer - A
  M 22:4 EmailNotificationStrategy.envoyer - A
  M 26:4 EmailNotificationStrategy.configurer - A
  M 31:4 EmailNotificationStrategy.envoyer_email - A
  M 40:4 SMSNotificationStrategy.envoyer - A
  M 44:4 SMSNotificationStrategy.configurer - A
  M 49:4 SMSNotificationStrategy.envoyer_sms - A
  M 58:4 PushNotificationStrategy.envoyer - A
  M 62:4 PushNotificationStrategy.configurer - A
  M 67:4 PushNotificationStrategy.envoyer_push - A
  M 76:4 NotificationContext.__init__ - A
  M 80:4 NotificationContext.notifier - A

C:\Users\DELL\PycharmProjects\projet_PMQL>radon raw notification.py
notification.py
  LOC: 82
  LLOC: 52
  SLOC: 33
  Comments: 6
  Single comments: 25
  Multi: 0
  Blank: 24
  - Comment Stats
    (C % L): 7%
    (C % S): 18%
    (C + M % L): 7%

C:\Users\DELL\PycharmProjects\projet_PMQL>radon mi notification.py
notification.py - A

C:\Users\DELL\PycharmProjects\projet_PMQL>radon hal notification.py
notification.py:
  h1: 0
```

Projet.py

```
Sélection C:\Windows\System32\cmd.exe
C:\Users\DELL\PycharmProjects\projet_PMQL>radon cc tests\test_projet.py
tests\test_projet.py
C 7:0 TestProjet - A
M 10:4 TestProjet.setUp - A
M 28:4 TestProjet.test_ajouter_tache - A
M 45:4 TestProjet.test_ajouter_membre_equipe - A
M 51:4 TestProjet.test_ajouter_risque - A
M 57:4 TestProjet.test_ajouter_jalon - A
M 63:4 TestProjet.test_enregistrer_changement - A

C:\Users\DELL\PycharmProjects\projet_PMQL>radon mi tests\test_projet.py
tests\test_projet.py - A

C:\Users\DELL\PycharmProjects\projet_PMQL>radon raw tests\test_projet.py
tests\test_projet.py
LOC: 71
LLOC: 42
SLOC: 52
Comments: 0
Single comments: 8
Multi: 0
Blank: 11
- Comment Stats
  (C % L): 0%
  (C % S): 0%
  (C + M % L): 0%

C:\Users\DELL\PycharmProjects\projet_PMQL>radon hal tests\test_projet.py
tests\test_projet.py:
h1: 1
h2: 2
N1: 1
N2: 2
vocabulary: 3
length: 3
calculated_length: 2.0
volume: 4.754887502163469
difficulty: 0.5
effort: 2.3774437510817346
time: 0.1320802083934297
bugs: 0.0015849625007211565
```

CONCLUSION

Au terme de cette aventure, nous avons parcouru un voyage passionnant à travers le monde de la gestion de projet et du développement logiciel. À travers ce projet, nous avons pu démontrer notre capacité à concevoir, développer et mettre en œuvre une application de gestion de projet fonctionnelle et efficace.

Nous avons réussi à créer un système robuste qui permet la gestion complète des projets, depuis la définition des objectifs jusqu'à l'accomplissement des tâches, en passant par la gestion des ressources et la gestion des risques. En intégrant des fonctionnalités telles que la création et la gestion des tâches, la notification des membres de l'équipe, la gestion des risques et des changements, nous avons pu répondre efficacement aux besoins de gestion de projet modernes.

Tout au long de ce projet, nous avons été confrontés à divers défis techniques et conceptuels. Cependant, grâce à notre détermination, notre collaboration et notre résolution de problèmes, nous avons surmonté ces obstacles et avons réussi à livrer un produit final de haute qualité.

Pour l'avenir, il reste encore beaucoup de possibilités d'amélioration et d'expansion de ce projet. Nous pourrions explorer des fonctionnalités supplémentaires telles que la gestion des dépendances entre les tâches, l'optimisation des notifications et l'intégration de métriques de performance pour évaluer l'efficacité du projet.

En conclusion, ce projet a été une expérience enrichissante qui nous a permis d'approfondir nos connaissances en gestion de projet et en développement logiciel. Nous sommes fiers du produit que nous avons créé et nous sommes reconnaissants envers toutes les personnes qui ont contribué à sa réalisation. Ce projet représente notre engagement envers l'excellence et notre passion pour l'innovation dans le domaine de la gestion de projet.