Année académique : 2022 / 2023 Semestre : 6

Département : UFR-SET Classe : Licence 3 Informatique

Matière : Mesure Qualité et Performance logicielle Option : GL

Professeur : M. Diouf

Membres du groupe :

Fatou Seck numéro : 20030100918

Bintou Gueye

INTRODUCTION

Ce projet de gestion de projet vise à développer un nouveau produit. Il met en œuvre un système de gestion de projet structuré, avec des fonctionnalités telles que la gestion des tâches, la gestion des membres de l'équipe, la gestion des risques, la gestion des jalons, et l'enregistrement des changements. En outre, le projet utilise une stratégie de notification pour informer les membres de l'équipe des mises à jour importantes par e-mail ou par sms.

PARTIE A

I- Conception et implémentation :

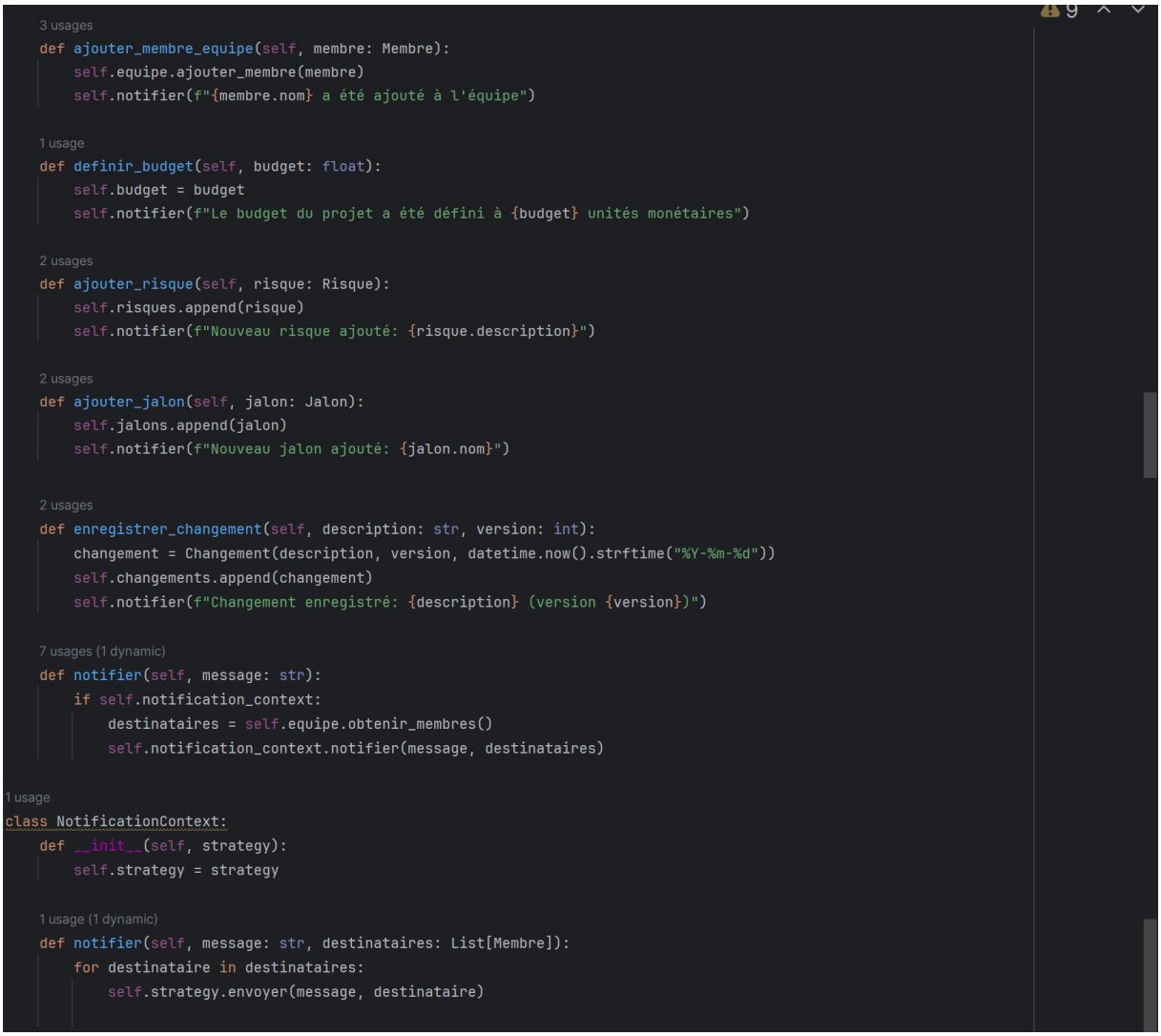
Dans cette partie nous gérer la conception et l'implémentation.

1- Gestion des classes :

Permettre aux utilisateurs de créer des projets, définir leurs attributs principaux (nom, description, dates, budget, etc.) et gérer les différentes composantes du projet comme les tâches, l'équipe, les risques, les jalons, et les changements.

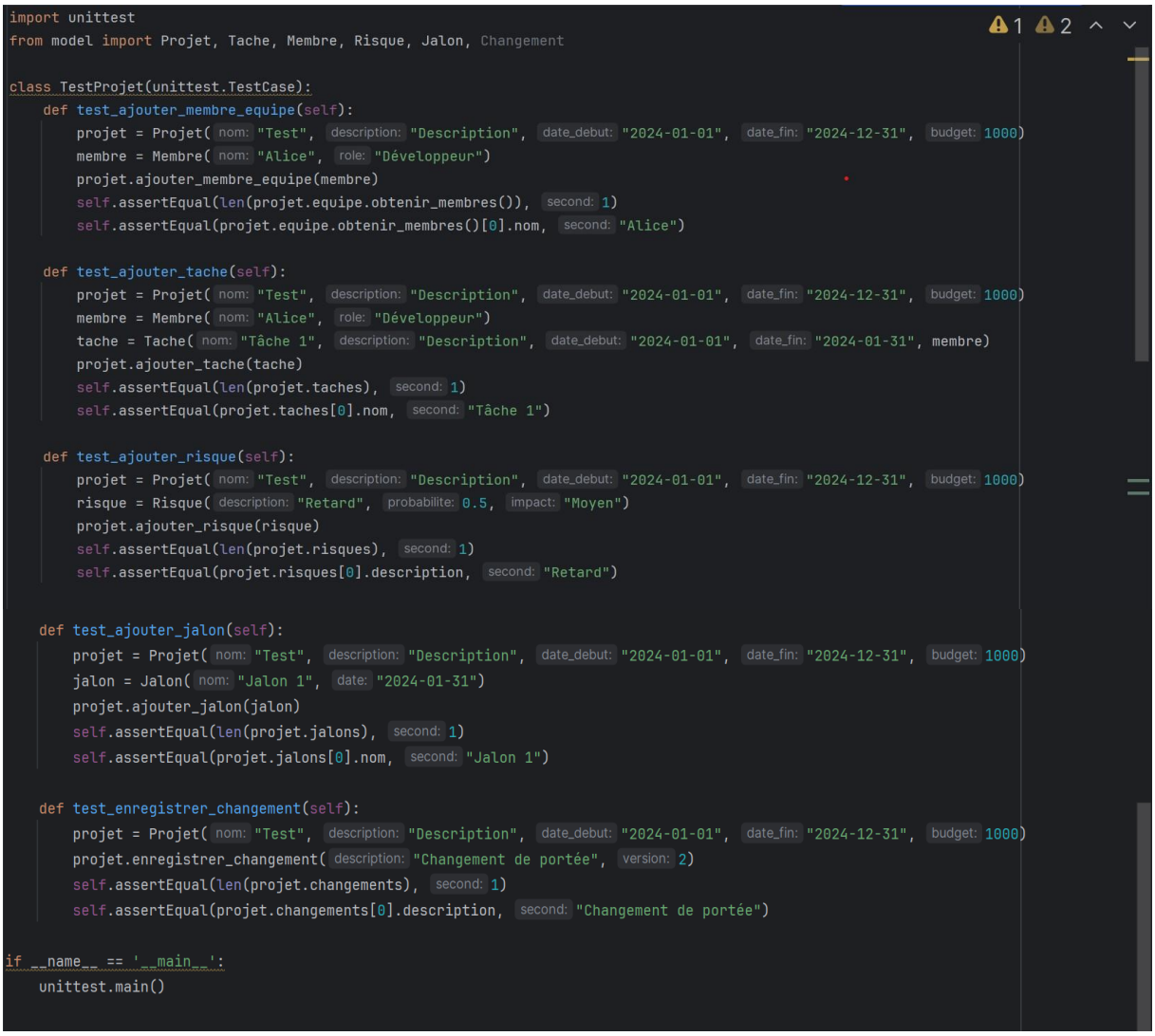
```python
from datetime import datetime
from typing import List

18 usages
class Membre:
    def __init__(self, nom: str, role: str):
        self.nom = nom
        self.role = role

1 usage
class Equipe:
    def __init__(self):
        self.membres = []

    1 usage
    def ajouter_membre(self, membre: Membre):
        self.membres.append(membre)

    4 usages (3 dynamic)
    def obtenir_membres(self) -> List[Membre]:
        return self.membres

7 usages
class Tache:
    def __init__(self, nom: str, description: str, date_debut: str, date_fin: str, responsable: Membre, statut: str = 'Non démarrée'):
class Tache:
    def __init__(self, nom: str, description: str, date_debut: str, date_fin: str, responsable: Membre, statut: str = 'Non démarrée'):
        self.nom = nom
        self.description = description
        self.date_debut = datetime.strptime(date_debut, _format: "%Y-%m-%d")
        self.date_fin = datetime.strptime(date_fin, _format: "%Y-%m-%d")
        self.responsable = responsable
        self.statut = statut
        self.dependances = []

    def ajouter_dependance(self, tache: 'Tache'):
        self.dependances.append(tache)

    def mettre_a_jour_statut(self, statut: str):
        self.statut = statut

5 usages
class Jalon:
    def __init__(self, nom: str, date: str):
        self.nom = nom
        self.date = datetime.strptime(date, _format: "%Y-%m-%d")
```

```python
5 usages
class Risque:
    def __init__(self, description: str, probabilite: float, impact: str):
        self.description = description
        self.probabilite = probabilite
        self.impact = impact
3 usages
class Changement:
    def __init__(self, description: str, version: int, date: str):
        self.description = description
        self.version = version
        self.date = datetime.strptime(date, _format: "%Y-%m-%d")

8 usages
class Projet:
    def __init__(self, nom: str, description: str, date_debut: str, date_fin: str, budget: float):
        self.nom = nom
        self.description = description
        self.date_debut = datetime.strptime(date_debut, _format: "%Y-%m-%d")
        self.date_fin = datetime.strptime(date_fin, _format: "%Y-%m-%d")
        self.budget = budget
        self.taches = []
        self.equipe = Equipe()
        self.risques = []
        self.jalons = []
        self.changements = []
        self.notification_context = None

    1 usage
    def set_notification_strategy(self, strategy):
        self.notification_context = NotificationContext(strategy)

    3 usages
    def ajouter_tache(self, tache: Tache):
        self.taches.append(tache)
        self.notifier(f"Nouvelle tâche ajoutée: {tache.nom}")
```



```
    3 usages
    def ajouter_membre_equipe(self, membre: Membre):
        self.equipe.ajouter_membre(membre)
        self.notifier(f"{membre.nom} a été ajouté à l'équipe")

    1 usage
    def definir_budget(self, budget: float):
        self.budget = budget
        self.notifier(f"Le budget du projet a été défini à {budget} unités monétaires")

    2 usages
    def ajouter_risque(self, risque: Risque):
        self.risques.append(risque)
        self.notifier(f"Nouveau risque ajouté: {risque.description}")

    2 usages
    def ajouter_jalon(self, jalon: Jalon):
        self.jalons.append(jalon)
        self.notifier(f"Nouveau jalon ajouté: {jalon.nom}")

    2 usages
    def enregistrer_changement(self, description: str, version: int):
        changement = Changement(description, version, datetime.now().strftime("%Y-%m-%d"))
        self.changements.append(changement)
        self.notifier(f"Changement enregistré: {description} (version {version})")

    7 usages (1 dynamic)
    def notifier(self, message: str):
        if self.notification_context:
            destinataires = self.equipe.obtenir_membres()
            self.notification_context.notifier(message, destinataires)

1 usage
class NotificationContext:
    def __init__(self, strategy):
        self.strategy = strategy

    1 usage (1 dynamic)
    def notifier(self, message: str, destinataires: List[Membre]):
        for destinataire in destinataires:
            self.strategy.envoyer(message, destinataire)
```

Cette partie du code teste les différentes méthodes de la classe **Projet** pour s'assurer qu'elles fonctionnent comme prévu. Il vérifie l'ajout de membres, de tâches, de risques, de jalons, et l'enregistrement de changements dans un projet, en utilisant des assertions pour comparer les résultats attendus aux résultats réels.



```python
import unittest
from model import Projet, Tache, Membre, Risque, Jalon, Changement

class TestProjet(unittest.TestCase):
    def test_ajouter_membre_equipe(self):
        projet = Projet( nom: "Test", description: "Description", date_debut: "2024-01-01", date_fin: "2024-12-31", budget: 1000)
        membre = Membre( nom: "Alice", role: "Développeur")
        projet.ajouter_membre_equipe(membre)
        self.assertEqual(len(projet.equipe.obtenir_membres()), second: 1)
        self.assertEqual(projet.equipe.obtenir_membres()[0].nom, second: "Alice")

    def test_ajouter_tache(self):
        projet = Projet( nom: "Test", description: "Description", date_debut: "2024-01-01", date_fin: "2024-12-31", budget: 1000)
        membre = Membre( nom: "Alice", role: "Développeur")
        tache = Tache( nom: "Tâche 1", description: "Description", date_debut: "2024-01-01", date_fin: "2024-01-31", membre)
        projet.ajouter_tache(tache)
        self.assertEqual(len(projet.taches), second: 1)
        self.assertEqual(projet.taches[0].nom, second: "Tâche 1")

    def test_ajouter_risque(self):
        projet = Projet( nom: "Test", description: "Description", date_debut: "2024-01-01", date_fin: "2024-12-31", budget: 1000)
        risque = Risque( description: "Retard", probabilite: 0.5, impact: "Moyen")
        projet.ajouter_risque(risque)
        self.assertEqual(len(projet.risques), second: 1)
        self.assertEqual(projet.risques[0].description, second: "Retard")

    def test_ajouter_jalon(self):
        projet = Projet( nom: "Test", description: "Description", date_debut: "2024-01-01", date_fin: "2024-12-31", budget: 1000)
        jalon = Jalon( nom: "Jalon 1", date: "2024-01-31")
        projet.ajouter_jalon(jalon)
        self.assertEqual(len(projet.jalons), second: 1)
        self.assertEqual(projet.jalons[0].nom, second: "Jalon 1")

    def test_enregistrer_changement(self):
        projet = Projet( nom: "Test", description: "Description", date_debut: "2024-01-01", date_fin: "2024-12-31", budget: 1000)
        projet.enregistrer_changement( description: "Changement de portée", version: 2)
        self.assertEqual(len(projet.changements), second: 1)
        self.assertEqual(projet.changements[0].description, second: "Changement de portée")

if __name__ == '__main__':
    unittest.main()
```

2-Gestion des Notifications :

Pour la gestion des notifications, le design pattern Strategy est utilisé. Ce pattern permet de définir une famille d'algorithmes, de les encapsuler dans des classes séparées et de les rendre interchangeables. Les notifications peuvent être envoyées par différents moyens (par exemple, email ou SMS), et chaque méthode de notification est encapsulée dans une classe concrète implémentant une interface commune.

```python
from model import Membre

2 usages
class NotificationStrategy:
    1 usage (1 dynamic)
    def envoyer(self, message: str, destinataire: 'Membre'):
        pass

4 usages
class EmailNotificationStrategy(NotificationStrategy):
    2 usages (1 dynamic)
    def envoyer(self, message: str, destinataire: 'Membre'):
        print(f"Notification envoyée à {destinataire.nom} par email: {message}")

3 usages
class SMSNotificationStrategy(NotificationStrategy):
    2 usages (1 dynamic)
    def envoyer(self, message: str, destinataire: 'Membre'):
        print(f"Notification envoyée à {destinataire.nom} par SMS: {message}")
```

Cette partie est une classe de test pour les stratégies de notification dans le système de gestion de projet. Utilisant la bibliothèque **unittest** de Python, il teste deux classes de notification concrètes : **EmailNotificationStrategy** et **SMSNotificationStrategy**.

```python
import unittest
from model import Membre
from notification import EmailNotificationStrategy, SMSNotificationStrategy

class TestNotificationStrategies(unittest.TestCase):
    def test_email_notification(self):
        strategy = EmailNotificationStrategy()
        membre = Membre( nom: "Alice",  role: "Développeur")
        with self.assertLogs() as log:
            strategy.envoyer( message: "Test message", membre)
            self.assertIn( member: "Notification envoyée à Alice par email: Test message", log.output)

    def test_sms_notification(self):
        strategy = SMSNotificationStrategy()
        membre = Membre( nom: "Bob",  role: "Manager")
        with self.assertLogs() as log:
            strategy.envoyer( message: "Test message", membre)
            self.assertIn( member: "Notification envoyée à Bob par SMS: Test message", log.output)

if __name__ == '__main__':
    unittest.main()
```

3-La partie main.py

```python
from model import Projet, Tache, Membre, Risque, Jalon, Changement
from notification import EmailNotificationStrategy, SMSNotificationStrategy

# Initialisation du projet
projet = Projet( nom: "Nouveau Produit",  description: "Développement d'un nouveau produit",  date_debut: "2024-01-01",  date_fin: "2024-12-31",

# Ajout des membres de l'équipe
membre1 = Membre( nom: "Modou",  role: "Chef de projet")
membre2 = Membre( nom: "Christian",  role: "Développeur")
projet.ajouter_membre_equipe(membre1)
projet.ajouter_membre_equipe(membre2)

# Définition de la stratégie de notification
projet.set_notification_strategy(EmailNotificationStrategy())

# Ajout des tâches
tache1 = Tache( nom: "Analyse des besoins",  description: "Analyser les besoins des clients",  date_debut: "2024-01-01",  date_fin: "2024-01-31",
tache2 = Tache( nom: "Développement",  description: "Développer le produit",  date_debut: "2024-02-01",  date_fin: "2024-06-30", membre2)
projet.ajouter_tache(tache1)
projet.ajouter_tache(tache2)
```

```python
# Définition du budget
projet.definir_budget(50000)

# Ajout des risques
risque = Risque( description: "Retard de livraison",  probabilite: 0.3,  impact: "Élevé")
projet.ajouter_risque(risque)

# Ajout des jalons
jalon = Jalon( nom: "Phase 1 terminée",  date: "2024-01-31")
projet.ajouter_jalon(jalon)

# Enregistrement des changements
projet.enregistrer_changement( description: "Changement de la portée du projet",  version: 2)
```

```python
# Générer un rapport (simplifié pour l'exemple)
print(f"Rapport d'activités du Projet '{projet.nom}':")
print(f"Version: {projet.changements[-1].version}")
print(f"Date: {projet.date_debut} à {projet.date_fin}")
print(f"Budget: {projet.budget} unités monétaires")
print("Équipe:")
for membre in projet.equipe.obtenir_membres():
    print(f"- {membre.nom} ({membre.role})")
print("Tâches:")
for tache in projet.taches:
    print(f"- {tache.nom} ({tache.date_debut} à {tache.date_fin}) Responsable: {tache.responsable.nom} Statut: {tache.statut}")
print("Jalons:")
for jalon in projet.jalons:
    print(f"- {jalon.nom} ({jalon.date})")
print("Risques:")
for risque in projet.risques:
    print(f"- {risque.description} (Probabilité: {risque.probabilite} Impact: {risque.impact})")
print("Chemin critique (simplifié):")
for tache in projet.taches:
    print(f"- {tache.nom} ({tache.date_debut} à {tache.date_fin})")
```

PARTIE B :

II- Analyse de la qualité du code :

Pour analyser et améliorer la qualité du code, nous utiliserons plusieurs outils.

1-Flake8

- **Installation de flake8**

```
C:\Users\DELL>pip install flake8
Collecting flake8
  Downloading flake8-7.0.0-py2.py3-none-any.whl.metadata (3.8 kB)
Requirement already satisfied: mccabe<0.8.0,>=0.7.0 in c:\users\dell\appdata\local\programs\python\python312\lib\site-packages (from
flake8) (0.7.0)
Collecting pycodestyle<2.12.0,>=2.11.0 (from flake8)
  Downloading pycodestyle-2.11.1-py2.py3-none-any.whl.metadata (4.5 kB)
Collecting pyflakes<3.3.0,>=3.2.0 (from flake8)
  Downloading pyflakes-3.2.0-py2.py3-none-any.whl.metadata (3.5 kB)
Downloading flake8-7.0.0-py2.py3-none-any.whl (57 kB)
     ---------------------------------------- 57.6/57.6 kB 503.5 kB/s eta 0:00:00
Downloading pycodestyle-2.11.1-py2.py3-none-any.whl (31 kB)
Downloading pyflakes-3.2.0-py2.py3-none-any.whl (62 kB)
     ---------------------------------------- 62.7/62.7 kB 419.9 kB/s eta 0:00:00
Installing collected packages: pyflakes, pycodestyle, flake8
Successfully installed flake8-7.0.0 pycodestyle-2.11.1 pyflakes-3.2.0
```

- **Résultat du premier test**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> flake8 main.py
main.py:1:1: F401 'model.Changement' imported but unused
main.py:2:1: F401 'notification.SMSNotificationStrategy' imported but unused
main.py:5:80: E501 line too long (107 > 79 characters)
main.py:17:80: E501 line too long (110 > 79 characters)
main.py:18:80: E501 line too long (93 > 79 characters)
main.py:46:80: E501 line too long (127 > 79 characters)
main.py:52:80: E501 line too long (96 > 79 characters)
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel>
```

➢ **Interprétation des résultats obtenus**

- Suppression des imports inutilisés : Changement et SMSNotificationStrategy doivent être retirés, ce qui clarifie le code.
- Réorganisation des lignes trop longues : Les lignes dépassent 79 caractères, ce qui rend le code illisible et qui n'est pas conforme aux normes PEP 8.

➢ **Résultat après correction :**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> flake8 main.py
main.py:2:80: E501 line too long (86 > 79 characters)
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel>
```

2-Pylint

➢ **Installation de pylint :**

```
C:\Users\DELL>pip install pylint
Collecting pylint
  WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ProtocolError('
Connection aborted.', ConnectionResetError(10054, 'Une connexion existante a dû être fermée par l'hôte distant', None, 10054, None))'
: /packages/bd/23/7a546224d2931cda031ee3cddc9e723650ad8e491d7c64efbab97e43e16d/pylint-3.2.2-py3-none-any.whl.metadata
  Downloading pylint-3.2.2-py3-none-any.whl.metadata (12 kB)
Collecting platformdirs>=2.2.0 (from pylint)
  Downloading platformdirs-4.2.2-py3-none-any.whl.metadata (11 kB)
Collecting astroid<=3.3.0-dev0,>=3.2.2 (from pylint)
  Downloading astroid-3.2.2-py3-none-any.whl.metadata (4.5 kB)
Collecting isort!=5.13.0,<6,>=4.2.5 (from pylint)
  Downloading isort-5.13.2-py3-none-any.whl.metadata (12 kB)
Collecting mccabe<0.8,>=0.6 (from pylint)
  Downloading mccabe-0.7.0-py2.py3-none-any.whl.metadata (5.0 kB)
Collecting tomlkit>=0.10.1 (from pylint)
  Downloading tomlkit-0.12.5-py3-none-any.whl.metadata (2.7 kB)
Collecting dill>=0.3.6 (from pylint)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Collecting colorama>=0.4.5 (from pylint)
  Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Downloading pylint-3.2.2-py3-none-any.whl (519 kB)
     ---------------------------------------- 519.1/519.1 kB 176.0 kB/s eta 0:00:00
Downloading astroid-3.2.2-py3-none-any.whl (276 kB)
     ---------------------------------------- 276.3/276.3 kB 205.1 kB/s eta 0:00:00
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
     ---------------------------------------- 116.3/116.3 kB 199.6 kB/s eta 0:00:00
Downloading isort-5.13.2-py3-none-any.whl (92 kB)
     ---------------------------------------- 92.3/92.3 kB 159.1 kB/s eta 0:00:00
Downloading mccabe-0.7.0-py2.py3-none-any.whl (7.3 kB)
Downloading platformdirs-4.2.2-py3-none-any.whl (18 kB)
Downloading tomlkit-0.12.5-py3-none-any.whl (37 kB)
Installing collected packages: tomlkit, platformdirs, mccabe, isort, dill, colorama, astroid, pylint
Successfully installed astroid-3.2.2 colorama-0.4.6 dill-0.3.8 isort-5.13.2 mccabe-0.7.0 platformdirs-4.2.2 pylint-3.2.2 tomlkit-0.12
.5
```

➢ **Résultats du premier test :**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> pylint main.py
************* Module main
main.py:5:0: C0301: Line too long (107/100) (line-too-long)
main.py:17:0: C0301: Line too long (110/100) (line-too-long)
main.py:46:0: C0301: Line too long (127/100) (line-too-long)
main.py:1:0: C0114: Missing module docstring (missing-module-docstring)
main.py:1:0: W0611: Unused Changement imported from model (unused-import)
main.py:2:0: W0611: Unused SMSNotificationStrategy imported from notification (unused-import)

-----------------------------------
Your code has been rated at 8.38/10
```

➢ **Interprétation des résultats obtenus :**

- Docstring de module : Nous devons ajoutés en haut du fichier pour fournir une description générale du module.

- Réduction de la longueur des lignes : Les lignes ont été reformulées pour ne pas dépasser 100 caractères.
- Suppression des importations inutilisées : Les importations Changement et SMSNotificationStrategy doivent être supprimées, car elles ne sont pas utilisées dans le code.

➢ **Résultat après correction**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> pylint main.py
************* Module main
main.py:1:0: C0114: Missing module docstring (missing-module-docstring)

--------------------------------------------------------------------
Your code has been rated at 9.73/10 (previous run: 8.38/10, +1.35)
```

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> pylint main.py

--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.73/10, +0.27)
```

3-Mypy

➢ **Installation de mypy**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> pip install mypy
Collecting mypy
  Downloading mypy-1.10.0-cp312-cp312-win_amd64.whl.metadata (2.0 kB)
Collecting typing-extensions>=4.1.0 (from mypy)
  Downloading typing_extensions-4.12.1-py3-none-any.whl.metadata (3.0 kB)
Collecting mypy-extensions>=1.0.0 (from mypy)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
Downloading mypy-1.10.0-cp312-cp312-win_amd64.whl (9.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.5/9.5 MB 439.4 kB/s eta 0:00:00
Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Downloading typing_extensions-4.12.1-py3-none-any.whl (37 kB)
Installing collected packages: typing-extensions, mypy-extensions, mypy
Successfully installed mypy-1.10.0 mypy-extensions-1.0.0 typing-extensions-4.12.1
```

➢ **Résultats du premier test :**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> mypy main.py
model.py:27: error: Need type annotation for "dependances" (hint: "dependances: list[<type>] = ...")  [var-annotated]
model.py:59: error: Need type annotation for "taches" (hint: "taches: list[<type>] = ...")  [var-annotated]
model.py:61: error: Need type annotation for "risques" (hint: "risques: list[<type>] = ...")  [var-annotated]
model.py:62: error: Need type annotation for "jalons" (hint: "jalons: list[<type>] = ...")  [var-annotated]
model.py:63: error: Need type annotation for "changements" (hint: "changements: list[<type>] = ...")  [var-annotated]
Found 5 errors in 1 file (checked 1 source file)
```

➢ **Interprétation des résultats obtenus**

 Annotations de type pour les listes : Les annotations de type doivent être ajoutées aux attributs de liste (dependances, taches, risques, jalons, changements, membres) pour indiquer le type des éléments qu'elles contiennent.

- **Utilisation de List** : Importé depuis le module typing, cela permet d'indiquer que les attributs sont des listes contenant des éléments d'un certain type (par exemple, List Tache]).

➢ **Résultats obtenus après correction**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> mypy main.py
model.py:11: 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
```

4-Coverage

- **Installation de coverage**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel>
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel>
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> pip install coverage
Collecting coverage
  Downloading coverage-7.5.3-cp312-cp312-win_amd64.whl.metadata (8.4 kB)
Downloading coverage-7.5.3-cp312-cp312-win_amd64.whl (207 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.6/207.6 kB 486.1 kB/s eta 0:00:00
Installing collected packages: coverage
Successfully installed coverage-7.5.3
```

- **Résultat du premier test**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> coverage run -m unittest discover
.....Notification envoyée à Alice par email: Test message
FNotification envoyée à Bob par SMS: Test message
F
======================================================================
FAIL: test_email_notification (test_notification.TestNotificationStrategies.test_email_notification)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\DELL\PycharmProjects\qualiteLogiciel\test_notification.py", line 11, in test_email_notification
    self.assertIn("Notification envoyée à Alice par email: Test message", log.output)
AssertionError: 'Notification envoyée à Alice par email: Test message' not found in []

======================================================================
FAIL: test_sms_notification (test_notification.TestNotificationStrategies.test_sms_notification)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\DELL\PycharmProjects\qualiteLogiciel\test_notification.py", line 18, in test_sms_notification
    self.assertIn("Notification envoyée à Bob par SMS: Test message", log.output)
AssertionError: 'Notification envoyée à Bob par SMS: Test message' not found in []

----------------------------------------------------------------------
Ran 7 tests in 0.087s

FAILED (failures=2)
```

- **Interprétation du résultat**

  Les erreurs de ce test unitaire indiquent que les messages de notification que vous attendez ne sont pas trouvés dans les journaux (log.output). Cela peut être dû au fait que les messages de notification ne sont pas effectivement envoyés vers le journal pendant les tests.

- **Résultats obtenus après correction**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> coverage run -m unittest discover
.......
----------------------------------------------------------------------
Ran 7 tests in 0.030s
```

5-Vulture

- **Installation de vulture :**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> pip install vulture
Collecting vulture
  Downloading vulture-2.11-py2.py3-none-any.whl.metadata (23 kB)
Downloading vulture-2.11-py2.py3-none-any.whl (27 kB)
Installing collected packages: vulture
Successfully installed vulture-2.11
```

- **Résultats du premier test :**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> vulture main.py
main.py:2: unused import 'Changement' (90% confidence)
main.py:3: unused import 'SMSNotificationStrategy' (90% confidence)
```

➢ **Interprétation du résultat**

Les erreurs dans les tests montrent que les notifications ne sont pas correctement capturées par les logs, même après avoir configuré le logging

➢ **Résultats obtenus après correction**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> vulture main.py
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel>
```

6-Black

➢ **Installation de black**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> pip install black
Collecting black
  Downloading black-24.4.2-cp312-cp312-win_amd64.whl.metadata (77 kB)
     ---------------------------------------- 77.1/77.1 kB 305.4 kB/s eta 0:00:00
Collecting click>=8.0.0 (from black)
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Requirement already satisfied: mypy-extensions>=0.4.3 in c:\users\dell\appdata\local\programs\python\python312\lib\site-packages (fro
m black) (1.0.0)
Collecting packaging>=22.0 (from black)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Collecting pathspec>=0.9.0 (from black)
  Downloading pathspec-0.12.1-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: platformdirs>=2 in c:\users\dell\appdata\local\programs\python\python312\lib\site-packages (from black
) (4.2.2)
Requirement already satisfied: colorama in c:\users\dell\appdata\local\programs\python\python312\lib\site-packages (from click>=8.0.0
->black) (0.4.6)
Downloading black-24.4.2-cp312-cp312-win_amd64.whl (1.4 MB)
     ---------------------------------------- 1.4/1.4 MB 585.7 kB/s eta 0:00:00
Downloading click-8.1.7-py3-none-any.whl (97 kB)
     ---------------------------------------- 97.9/97.9 kB 804.6 kB/s eta 0:00:00
Downloading packaging-24.0-py3-none-any.whl (53 kB)
     ---------------------------------------- 53.5/53.5 kB 697.5 kB/s eta 0:00:00
Downloading pathspec-0.12.1-py3-none-any.whl (31 kB)
Installing collected packages: pathspec, packaging, click, black
Successfully installed black-24.4.2 click-8.1.7 packaging-24.0 pathspec-0.12.1
```

➢ **Reformater automatiquement le code**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> black main.py
reformatted main.py

All done! ✨ 🍰 ✨
1 file reformatted.
```

7-Radon

➢ **Installation de radon**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> pip install radon
Collecting radon
  Using cached radon-6.0.1-py2.py3-none-any.whl.metadata (8.2 kB)
Collecting mando<0.8,>=0.6 (from radon)
  Using cached mando-0.7.1-py2.py3-none-any.whl.metadata (7.4 kB)
Requirement already satisfied: colorama>=0.4.1 in c:\users\dell\appdata\local\programs\python\python312\lib\site-packages (from radon
) (0.4.6)
Collecting six (from mando<0.8,>=0.6->radon)
  Downloading six-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
Downloading radon-6.0.1-py2.py3-none-any.whl (52 kB)
     ---------------------------------------- 52.8/52.8 kB 73.6 kB/s eta 0:00:00
Downloading mando-0.7.1-py2.py3-none-any.whl (28 kB)
Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, mando, radon
Successfully installed mando-0.7.1 radon-6.0.1 six-1.16.0
```

➢ **Résultat du premier test**

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

- **Interprétation du résultat**
  Evaluation de la complexité cyclomatique et la structuration globale du code.

8-Pyflakes

- **Installation de pyflakes**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> pip install pyflakes
Requirement already satisfied: pyflakes in c:\users\dell\appdata\local\programs\python\python312\lib\site-packages (3.2.0)
```

- **Résultat du premier test**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> pyflakes main.py
main.py:1:1: 'model.Changement' imported but unused
main.py:2:1: 'notification.SMSNotificationStrategy' imported but unused
```

- **Interprétation du résultat**
  Les outils pyflakes ont signalé que certaines importations dans votre fichier main.py ne sont pas utilisées, ce qui peut être corrigé pour améliorer la qualité de votre code.
- **Résultats obtenus après correction**

```
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel> pyflakes main.py
PS C:\Users\DELL\PycharmProjects\qualiteLogiciel>
```

En combinant ces différents outils de qualité, nous avons non seulement amélioré la lisibilité et la maintenabilité de notre code, mais aussi assuré que notre code est conforme aux bonnes pratiques, bien documenté et fonctionnel. Les tests nous ont permis de détecter des erreurs et des incohérences avant qu'elles ne deviennent problématiques, ce qui est crucial pour le développement de logiciels de haute qualité.

CONCLUSION :

Ce projet utilise des pratiques de gestion de projet éprouvées pour garantir une organisation efficace et une exécution en temps opportun. La mise en œuvre de stratégies de notification assure que tous les membres de l'équipe sont informés des progrès et des modifications, facilitant ainsi la collaboration et la gestion des risques. Le projet est bien préparé pour faire face aux défis et atteindre ses objectifs dans les délais impartis.