Guide de développement

Décembre 2023 – Version : 1.0

Antoine TASSEZ

Introduction

Ce document a pour objectif de définir les normes, les règles et les meilleures pratiques à suivre lors du développement du projet. L'adoption de ces directives vise à garantir la cohérence, la lisibilité et la qualité du code, facilitant ainsi la collaboration au sein du projet. Ce document couvre plusieurs aspects essentiels du processus de développement, y compris le nommage des éléments, l'utilisation des commentaires et des docstrings, la mise en place de tests unitaires et d'intégration, ainsi que l'organisation des branches et les règles de push.

Table des matières

1. Nommage des Classes, Variables, Méthodes, Fonctions et Constantes

1.1 Classes

Pour le nommage des classes :

- Le projet utilise la notation CamelCase.
- Les noms choisis sont descriptifs, reflétant la nature de la classe.

- Les abréviations non explicites sont à proscrire.

Exemple :

```
class MaClasseImportante:
    # Corps de la classe
```

1.2 Variables, Méthodes et fonctions

Pour le nommage des variables, méthodes et fonctions :

- Le projet utilise la notation snake_case.
- Les noms choisis sont descriptifs, reflétant la nature de la variable ou l'implémentation de la méthode/fonction.
- Les abréviations non explicites sont à proscrire.

Exemple :

```
variable_importante = 42

def methode_exemplaire():
    # Corps de la méthode
```

1.3 Constantes

Pour le nommage des constantes :

- Le projet utilise la notation SNAKE_CASE_MAJ.
- Les noms choisis sont descriptifs, reflétant la nature de la constante.

Exemple :

```
CONSTANTE_IMPORTANTE = 3.14
```

Note : Les constantes sont des variables dont la valeur ne devrait pas changer pendant l'exécution du programme. Assurez-vous que la valeur d'une constante est en majuscules pour indiquer clairement son statut.

2. Commentaires

2.1 En-tête de Fichier

- Incluez une brève description du fichier, le nom de l'auteur et la date de création.

Exemple :

```
"""
Fichier : mon_module.py
Auteur : Antoine
Date de création : 12-2023
"""
```

2.2 Commentaires en Ligne

- Des commentaires doivent être présents pour expliquer des portions de code complexes ou non évidentes.
- Les commentaires doivent être clairs et concis.

3. DocString

Les docstrings doivent être écrites à la création de la classe ou fonction et mises à jour lors de chaque itération sur le code afin de s’assurer que les commentaires sont toujours force de vérité.

3.1 Classes et Fonctions

- Incluez une docstring pour chaque classe et fonction.
- Décrivez l'objectif, les paramètres et la valeur de retour le cas échéant.

Exemple :

```
def ma_fonction(parametre):
    """
    Description de la fonction.

    Args:
        parametre (type): Description du paramètre.

    Returns:
        type: Description de la valeur de retour.
    """
    # Corps de la fonction
```

4. Tests Unitaires et d'Intégration

Les tests doivent être écrits lors du développement de toutes nouvelles fonctionnalités. Le test d’une nouvelle fonction doit être mis à disposition sur le dépôt avant la fonction.

4.1 Test Unitaires

- Écrivez des tests pour chaque fonction ou méthode.
- Un minimum de 80 % de couverture du code par les tests est attendu.
- Les tests doivent être autonomes et reproductibles.

4.2 Tests d'Intégration

- Vérifiez l'intégration entre les différents composants du système.
- Testez les scénarios réalistes de bout en bout. Les Happy paths doivent aussi être testés.

5. Organisation des Branches et Règles de Push

5.1 Nommage des Branches

- Utilisez une convention de nommage claire, comme feature/nom_de_la_fonctionnalite.

- Chaque sprint doit posséder sa propre branche.
- Avant le merge d’une branche de sprint sur la branche de développement, une branche QA doit être faite.

5.2 Règles de Push

- Ne poussez sur la branche principale (par exemple, main ou master) qu'après une revue de code.
- Seule la branche « Dévelop » peut être mergée avec la branche principale. Tout développement doit être fait sur une branche adaptée (sprint_0 par ex.), avant d’être mergé sur la branche develop avec les tests appropriés.