POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

CODE DE QUALITÉ

28 AOÛT 2023

CHARLES DE LAFONTAINE, JÉRÔME COLLIN, MERIAM BEN RABIA
GIGL | Polytechnique Montréal

Ce document est protégé par les droits d'auteurs en vertu de la licence Creative Commons Attribution 4.0 International (**CC BY 4.0**). Vous êtes autorisé(e) à partager, copier, distribuer et communiquer au public ce document, à condition d'attribuer correctement la paternité en citant les auteurs originaux. Vous n'êtes pas autorisé(e) à utiliser ce document à des fins commerciales. Toute modification de ce document doit être clairement indiquée, et les nouvelles créations doivent être diffusées sous une licence similaire.

N.B. Le masculin est utilisé pour alléger le texte.

TABLE DES MATIÈRES

CODE DE QUALITÉ

La programmation, que vous soyez nouveau dans le domaine ou un vétéran chevronné, reste un domaine à la fois fascinant et complexe. Peu importe où vous en êtes dans votre parcours, une chose reste essentielle : l'importance de produire un code de qualité. C'est une compétence qui distingue un bon ingénieur logiciel d'un grand ingénieur logiciel. Mais qu'est-ce qu'un « code de qualité » exactement ? Et comment pouvez-vous, en tant qu'ingénieur logiciel, vous assurer que votre code répond à cette norme ?

1. CE QUI DÉFINIT UN CODE DE QUALITÉ

Un code de qualité est un assemblage d'instructions qui non seulement fonctionne comme prévu, mais est aussi structuré de manière à être compréhensible et réutilisable par autrui. Plus particulièrement, votre code doit être :

1.1. Fonctionnel

Un code de qualité doit fonctionner correctement, c'est-à-dire qu'il doit accomplir la tâche pour laquelle il a été conçu sans générer d'erreurs ou de comportements inattendus (de bonne foi).

1.2. Lisible

Le code doit être facile à lire et à comprendre. Cela signifie que les noms des variables et des fonctions doivent être descriptifs, que l'indentation doit être correcte et que les conventions de nommage appropriées doivent être respectées.

1.3. Maintenable

Un code de qualité est conçu de manière à faciliter sa maintenance ultérieure. Cela signifie qu'il est facile de corriger les bogues, d'ajouter de nouvelles fonctionnalités et d'améliorer le code sans avoir à réécrire de grandes parties du code.

1.4. Modulaire

Un code modulaire est divisé en parties ou modules qui accomplissent des tâches spécifiques. Cela rend le code plus organisé, plus facile à comprendre et à tester.

1.5. Efficient

Un code de qualité doit être optimisé pour l'efficacité, ce qui signifie qu'il exécute les tâches aussi rapidement que possible avec une utilisation minimale des ressources.

1.6. Robuste

Un code de qualité doit être capable de gérer des situations imprévues ou des entrées incorrectes sans se bloquer ou produire des résultats incorrects.

Un code de qualité respecte les principes de développement de logiciels solides (*SOLID*), des pratiques de codage établies qui maximisent la maintenabilité et minimisent la complexité, et respecte les normes de codage établies, qui garantissent la cohérence et la lisibilité. En apprendre plus…

2. RECOMMANDATIONS GÉNÉRALES ET BONNES PRATIQUES

2.1. Lisibilité

La lisibilité d'un code est améliorée en utilisant des noms de variables, de fonctions et de classes descriptifs, qui donnent une indication claire de leur rôle ou de la tâche qu'ils accomplissent. Les commentaires peuvent également être utilisés pour fournir des explications supplémentaires sur des parties complexes du code. Par exemple, un bon commentaire pourrait expliquer pourquoi une certaine approche a été utilisée, ou décrire les effets secondaires d'une fonction.

2.2. Modularité

La modularité implique de diviser un programme en unités plus petites, appelées modules ou fonctions, qui accomplissent chacune une tâche spécifique. Par exemple, au lieu d'avoir une longue fonction qui fait plusieurs choses, il serait préférable d'avoir plusieurs petites fonctions qui font chacune une chose. La modularité facilite la compréhension du code, car chaque module peut être compris indépendamment des autres. Elle facilite également la réutilisation du code et rend le code plus facile à tester.

2.3. Simplicité

Le principe « *Keep it simple, stupid* » (*KISS*) exhorte les développeurs à éviter les solutions inutilement compliquées. Un code simple est généralement plus facile à comprendre et à maintenir qu'un code complexe. De plus, il est moins susceptible de contenir des bogues, car il y a moins d'endroits où des erreurs peuvent se cacher.

2.4. Réduire la duplication

La duplication de code se produit lorsque le même bloc de code est copié et collé à plusieurs endroits. Cela peut rendre le code plus difficile à maintenir, car une modification dans la logique du code pourrait nécessiter de changer le même code à plusieurs endroits. Le principe « *Don't Repeat Yourself* » (*DRY*) encourage les développeurs à éviter la duplication de code en extrayant le code commun en une fonction ou un module qui peut être appelé à plusieurs endroits.

2.5. Utiliser le contrôle de version

Les systèmes de contrôle de version comme *Git* permettent de suivre les modifications du code au fil du temps, de collaborer plus efficacement avec d'autres développeurs, et d'annuler ou de comparer les modifications si nécessaire.

3. BIENFAITS D'UN CODE DE QUALITÉ ET MÉFAITS D'UN CODE DE MOINDRE QUALITÉ

3.1. Bienfaits d'un code de qualité

Un code de qualité est plus facile à comprendre, ce qui permet de gagner du temps lors de l'ajout de nouvelles fonctionnalités ou de la correction de bogues. Il est également plus facile à maintenir, car les modifications sont plus prévisibles et ont moins de chances d'introduire des bogues. De plus, un code de qualité est plus facile à tester, car sa modularité permet de tester chaque partie individuellement. Enfin, un code de qualité facilite la collaboration, car il est plus facile pour les autres développeurs de comprendre et de modifier votre code.

3.2. Méfaits d'un code de moindre qualité

Un code de moindre qualité peut causer de nombreux problèmes. Il est souvent source de bogues, car les erreurs sont plus difficiles à repérer dans un code compliqué ou mal organisé. Il est également plus difficile à comprendre et à maintenir, ce qui peut ralentir considérablement le développement. En outre, un code de moindre qualité peut créer des problèmes de performance si le code n'est pas optimisé, ou des problèmes de sécurité si les bonnes pratiques de sécurité ne sont pas respectées.

4. OUTILS ET MÉTHODES POUR AMÉLIORER LA QUALITÉ DU CODE

4.1. Examen par les pairs

L'examen par les pairs implique de faire relire votre code par un autre développeur avant qu'il ne soit fusionné dans la branche principale du code. Cela peut aider à détecter les erreurs que vous avez pu manquer, à améliorer la lisibilité du code et à partager les connaissances avec vos coéquipiers. L'examen par les pairs peut également aider à maintenir la cohérence du code, car les pairs peuvent signaler les écarts par rapport aux normes de codage de l'équipe.

4.2. Pipelines d'intégration/déploiement continus (*CI/CD*)

Les pipelines *CI/CD* sont des processus automatisés qui construisent, testent et déploient votre code chaque fois que des modifications sont apportées. Ils peuvent aider à détecter les erreurs plus rapidement, car les tests sont exécutés automatiquement à chaque modification. Ils peuvent également faciliter le déploiement du code en production, car le processus de déploiement est également automatisé.

4.3. *Linter* et formateurs de code

Un *linter* est un outil qui analyse automatiquement votre code pour détecter les erreurs de syntaxe, les problèmes de style, et parfois les problèmes de performance ou de sécurité. Un formateur de code est un outil qui reformate automatiquement votre code pour qu'il respecte une certaine convention de style. Ces outils peuvent vous aider à maintenir la qualité de votre code en détectant et en corrigeant automatiquement les problèmes mineurs.

4.4. Tests automatisés

Les tests automatisés sont des scripts qui exécutent votre code et vérifient que son comportement est correct. Il existe différents types de tests, tels que les tests unitaires (qui testent une seule fonction ou module), les tests d'intégration (qui testent plusieurs modules ensemble) et les tests de bout en bout (qui testent le système entier, généralement à travers l'interface utilisateur). Les tests automatisés peuvent vous aider à détecter les erreurs plus rapidement, à améliorer la robustesse de votre code et à développer de nouvelles fonctionnalités avec plus de confiance.

4.5. Extensions utiles

Il existe de nombreuses extensions pour les éditeurs de code et les *IDEs* qui peuvent aider à améliorer la qualité du code. Par exemple, il existe des plugiciels de *linter*, de formatage automatique, de débogage, de gestion des versions *Git*, de détection, de duplication de code, etc. Ces extensions peuvent vous aider à écrire un meilleur code en vous donnant des retours en temps réel sur la qualité de votre code et en vous aidant à corriger les problèmes avant qu'ils ne deviennent des bogues (voir *Prettier*, *SonarLint*, *GitLens*).

5. POUR ALLER PLUS LOIN

Des références comme « *Clean Code : A Handbook of Agile Software Craftsmanship* » de Robert C. Martin, ou « *Code Complete : A Practical Handbook of Software Construction* » de Steve McConnell, sont d'excellentes sources pour approfondir les principes de qualité du code.

Un code de qualité n'est pas seulement un objectif en soi, mais aussi une compétence essentielle pour tout ingénieur logiciel en devenir. En adoptant les bonnes pratiques et en utilisant les bons outils, vous pouvez produire un code plus robuste, plus sûr et plus facile à maintenir, ce qui vous rendra plus précieux en tant qu'ingénieur logiciel, tout en rendant votre travail plus agréable.