Quy định Git workflow và quy ước code

nhasioc

TỔNG QUAN

I. Mục đích

Tài liệu này nhằm:

- Thống nhất cách quản lý source code trong dự án
- Giảm xung đột khi làm việc nhóm
- Đảm bảo code dễ đọc, dễ review, dễ bảo trì
- Làm căn cứ xử lý khi vi phạm quy trình kỹ thuật

Quy định này áp dụng cho toàn bộ thành viên tham gia phát triển và chỉnh sửa source code của dự án.

II. Nguyên tắc chung

1. Không làm việc trực tiếp trên nhánh chính

- Không commit trực tiếp vào main
- Không commit trực tiếp vào develop
- Mọi thay đổi phải thông qua nhánh riêng và Pull Request

Vi phạm quy định này được xem là vi phạm quy trình kỹ thuật.

---

2. Mọi thay đổi đều phải trace được

- Mỗi commit phải gắn với task hoặc issue cụ thể
- Không commit không rõ mục đích
- Không chỉnh sửa code mà không có liên quan đến task

Code không trace được nguồn gốc được xem là không hợp lệ.

---

3. Code phải ưu tiên tính rõ ràng

- Viết code để người khác đọc được
- Không tối ưu hóa sớm khi chưa cần thiết
- Không viết logic phức tạp nếu có thể tách nhỏ

Code khó đọc, khó hiểu sẽ bị yêu cầu chỉnh sửa trong quá trình review.

QUY ĐỊNH GIT WORKFLOW

III. Cấu trúc nhánh

Dự án sử dụng cấu trúc nhánh:

- main: nhánh production
- develop: nhánh tích hợp chung
- feature/*: phát triển tính năng
- bugfix/*: sửa lỗi
- hotfix/*: sửa lỗi khẩn cấp production

Không tạo nhánh tự do ngoài quy ước trên nếu chưa được thống nhất.

IV. Quy trình thực hiện task

4. Tạo nhánh

Nhánh phải được tạo từ develop:

- Luôn pull develop mới nhất trước khi tạo nhánh
- Không tạo nhánh từ nhánh cá nhân khác

Quy ước đặt tên nhánh:

- feature/TASKID-mo-ta-ngan
- bugfix/TASKID-mo-ta-ngan
- hotfix/TASKID-mo-ta-ngan

Yêu cầu:

- Không dùng tiếng Việt có dấu
- Không viết hoa tùy ý
- Không đặt tên chung chung như: fix, test, abc

---

5. Commit

Mỗi commit phải:

- Có nội dung rõ ràng
- Gắn với mã task
- Phản ánh đúng thay đổi thực tế

Cấu trúc commit message:

[TASK-ID] Loại thay đổi: Mô tả ngắn

Loại thay đổi bao gồm:

- Add
- Fix
- Update
- Refactor
- Remove
- Optimize
- Docs

Không chấp nhận commit message không có ý nghĩa.

---

6. Pull Request

Khi hoàn thành task:

- Push nhánh lên remote
- Tạo Pull Request vào develop
- Mô tả rõ:
  - Mục đích thay đổi
  - Phạm vi ảnh hưởng
  - Điểm cần lưu ý

Không tự ý merge khi chưa được review.

---

7. Review

Mỗi Pull Request phải:

- Có ít nhất 1 người review

- Được approve trước khi merge

Reviewer có trách nhiệm:

- Đọc và hiểu logic
- Kiểm tra ảnh hưởng hệ thống
- Yêu cầu chỉnh sửa nếu cần

Approve hình thức hoặc không đọc kỹ được xem là không hoàn thành trách nhiệm review.

---

8. Merge và xử lý conflict

- Người tạo PR chịu trách nhiệm xử lý conflict
- Phải test lại sau khi resolve conflict
- Không force push vào nhánh chung

Sau khi merge, người thực hiện task chịu trách nhiệm theo dõi lỗi phát sinh liên quan.

QUY ƯỚC CODE

V. Quy ước đặt tên

9. Biến

- Sử dụng camelCase
- Tên phải có ý nghĩa
- Không viết tắt khó hiểu

Không sử dụng tên như: a, data1, temp, test nếu không có ý nghĩa rõ ràng.

---

10. Class và Interface

- Sử dụng PascalCase
- Tên phản ánh đúng chức năng

Ví dụ:

- UserSyncService
- AuthController
- DataMappingRepository

Không đặt tên chung chung như: Service1, TestClass.

---

11. Tên file

- Tên file trùng với class chính
- Không đặt tên file không rõ nghĩa

Không sử dụng các tên như: utils2, newfile, test.

VI. Cấu trúc và tổ chức code

12. Phân tầng rõ ràng

- Controller không chứa business logic phức tạp
- Service không truy cập database trực tiếp nếu đã có repository layer
- Không viết toàn bộ logic trong một file duy nhất

Phải tuân thủ đúng kiến trúc đã thống nhất của dự án.

---

13. Hàm và logic

- Một function chỉ nên thực hiện một nhiệm vụ
- Không viết function quá dài và khó theo dõi
- Logic phức tạp phải được tách nhỏ

Nếu một function vượt quá mức hợp lý và khó đọc, reviewer có quyền yêu cầu refactor.

---

14. Logging

- Log phải có mục đích rõ ràng
- Không log dữ liệu nhạy cảm
- Không để log debug dư thừa trước khi merge

---

15. Xử lý lỗi

- Không catch lỗi mà bỏ qua
- Không throw lỗi chung chung
- Message lỗi phải rõ ràng, có ngữ cảnh

Không để các khối catch rỗng hoặc bỏ qua exception.

---

16. Format và kiểm tra tự động

- Bắt buộc sử dụng ESLint / Prettier theo cấu hình dự án
- Không merge nếu format sai
- Không thay đổi format toàn bộ file nếu không liên quan đến task

---

VII. Trách nhiệm kỹ thuật

1. Code trước khi tạo PR phải:
    - Chạy được
    - Không lỗi compile
    - Không làm hỏng chức năng cũ
2. Người thực hiện task chịu trách nhiệm:
    - Test lại trước khi merge
    - Theo dõi lỗi phát sinh sau khi merge
3. Không đổ lỗi cho quá trình merge nếu chưa kiểm tra kỹ trước đó.

VIII. Xử lý vi phạm

17. Vi phạm mức độ 1 – Yêu cầu chỉnh sửa

Áp dụng khi:

- Sai format
- Sai naming
- Commit message không đúng quy ước
- PR thiếu mô tả

Hình thức xử lý:

- Yêu cầu chỉnh sửa trước khi approve

---

18. Vi phạm mức độ 2 – Từ chối merge

Áp dụng khi:

- Không tuân thủ workflow
- Tự ý commit vào nhánh chính
- Không xử lý conflict
- Không test trước khi tạo PR

Hình thức xử lý:

- Từ chối merge
- Yêu cầu làm lại theo đúng quy trình

---

19. Vi phạm mức độ 3 – Đánh giá năng lực kỹ thuật

Áp dụng khi:

- Lặp lại vi phạm nhiều lần
- Không tuân thủ quy ước dù đã được nhắc
- Gây ảnh hưởng nghiêm trọng đến hệ thống

Hình thức xử lý:

- Trao đổi trực tiếp
- Đánh giá lại mức độ phù hợp trong dự án

IX. Hiệu lực

Quy định này có hiệu lực kể từ ngày ban hành và áp dụng cho toàn bộ thành viên tham gia phát triển dự án.

Mọi thay đổi về workflow hoặc quy ước kỹ thuật phải được thống nhất trước khi áp dụng.