Как коммитить и не быть убитым на почве ненависти

Nikita Sivakov

В моем мире страдающего перфекционизма коммиты — больная тема, вызывающая максимальные порывы ненависти. Попробую рассказать, как делать хорошо.

В каждом коммите должны проходить тесты

История должна быть такой, чтобы на любом коммите можно было запустить проект, а тесты выполнялись.

Соблюдение этого правила значительно упростит особо суровую отладку, позволит без проблем выбросить лишний коммит (иногда люди пишут код, а потом передумывают) и даст читателю уверенность, что это изменение — рабочее.

Это, наверное, самое лучшее правило, потому что ряд следующих требует его соблюдения.

Коммить тесты вместе с фичей

Читателю не придется искать тесты — он может

посмотреть историю текущего места и увидить тест (и наоборот).

Если код для решения задачи раскидан по проекту, то читатель сможет легко посмотреть только его, потому что только это было в коммите.

Иногда это не работает. Например, при значительных последующих изменениях кода можно наткнуться на другой коммит (вдруг кто-то массово рефакторил тесты), но в большинстве случаев правило будет хорошо работать.

Не коммить мелкие исправления текущей фичи отдельно

Грех, которым я долго страдал — коммиты в стиле “исправил очепятку”, “починил тест” и такое прочее. Такие коммиты не несут никакой ценности, засоряют историю и раздражают читателя — каждый злится, когда вызвав `blame` видит “fix typo” с исправлением буквы.

Если забыл что-то добавить в последний коммит, то просто сделай `--amend`. Забыл что-то добавить не в последний коммит — закоммить с текстом-напоминанием себе, а потом сделай `squash`.

Здесь бывают исключения —например, код ушел в продакшн, а автор только сейчас осознал ошибку. Упарываться переписыванием истории не стоит — есть риск ошибиться, что-то сломать и создать себе лишней

работы на вечер.

Один коммит — одна атомарная фича

Это сложно, но нужно стараться. Большие коммиты плохие, потому что сложно разбираться в коде, которого много. Автор посмотрит в изменения и ничего не поймет.

Очень грубый пример — нам нужно добавить флаг “мусор” к товарам, вывести это в апи, написать фильтры к апи по этому параметру, а еще создание\изменение\удаление товаров с этим флагом требуют особой не тривиальной логики и вообще тут **очень много кода**:

Плохо:

- работа с флагом “мусор” в апи товаров

Хорошо:

- вывел флаг мусор в апи товаров
- написал фильтрацию апи товаров по флагу “мусор”
- обработка создания товаров с флагом “мусор”
- обработки изменения флага “мусор” у товаров
- обработка удаления товаров с флагом “мусор”

Писать цель изменения

- PriceInputParser -> Написал парсер цен для обработки мусорного ввода

- Исправил дефолтные значения аргументов функции -> Исправил ошибку, из-за которой продакшн падал в каждое полнолуние
- Написал компонент SomeStuffSelector ->Написал переиспользуемый компонент выпадающего меню для выбор всяких штук

Коммить сразу

Моя частая проблема — я сначала напишу много кода, а потом начинаю стэйджить ханками и делать коммиты. Так легко ошибиться, а выделить маленькие изменения из кучи нового кода крайне сложно.

Хорошо коммитить по мере написания. Можно даже чаще, чем нужно — потом лишние легко перемещать и склеивать.

А еще это упростит работу над задачей, если сделал перерыв на несколько часов\дней и все забыл.

Писать детали — хорошо

Первая строка текста в коммите — его название. Все что после пустой строки под этим — его описание. Там можно изложить детальнее смысл своих изменений, указать ссылку на задачу и иную полезную иформацию.

Без фанатизма

У любого (почти) правила есть исключения.

Не нужно каждый раз переписывать всю историю и страдать полчаса над сообщением для коммита. Всегда важно следовать принятым в проекте правилам и закону о здравом смысле.