КАЧЕСТВО КОДА: БОРОТЬСЯ ИЛИ УЙТИ

ЧТО ЭТО

- Статья, набор утилит с конфигами, скриптов, шаблонов, примеров и наставлений
- Анализ в динамике: что мы исправили за год
- В основе 5 лет опыта работы в промышленных проектах на Angular
- Ссылка на статью со ссылками: https://gitlab.com/stepanovv/kbo/-/blob/master/public/articles/public/качество кода/качество_кода.md

ДЛЯ ЧЕГО ЭТО

- Ответ на вопрос - стоит ли бороться за проект или пора уходить
- Качество кода - главное объективное ограничение успеха проекта
- Анализ качества кода полезен для:
  - выхода проекта в open source
  - ежедневной борьбы за живучесть
  - планирования вех, ресурсов, размера команды
- Проблемы:
  - разработчики не хотят работать с плохим кодом
  - бюджет, сроки, SP, DevOPS, CI/CD
  - субъективность, чувства людей

Преимущества

- Проверено на разных типах проектов: react/angular/JS
- Проверено на больших проектах: 200к+ строк кода ts
- Более значимые/конкретные результаты в отличии от утилит типа codecov, codescene, sonar и типовых наборов правил линтеров
- Скорость анализа слабо/нелинейно зависит от размера проекта
- Не требуют настройки окружения/CI
- Нет внешних/облачных сервисов, которые могут сломаться после обновления версий или по желанию вендора
- Можно использовать по частям

ОГРАНИЧЕНИЯ

- это больше конструктор, чем готовое решение
- может быть трудно подобрать совместимые версии плагинов линтера и фреймворка
- работа скриптов проверена только для ОС Linux
- в монорепах/NX есть трудности со статическим анализом(ESLint)
- jscpd нужно настраивать для больших проектов
  - выключить blame
  - уменьшить порог срабатывания
  - проверять отдельно html|ts|css

КАК ИСПОЛЬЗОВАТЬ

- Скрипты разделены на две группы - для запуска в папке с утилитами и в папке проекта.
  - Интегрировать в проект может быть затруднительно технически/ организационно
- В шаблоне отчёта рядом со значениями есть скрипты-однострочники для выборочного запуска.
- Скрипты и утилиты можно подключить непосредственно в IDE.
- Результат работы скриптов - папка с логами и отчётами markdown.
- Данные из отчётов заносятся вручную в шаблон. Информация агрегирована/подготовлена к копированию.

ОТЧЁТ

- Разбит на разделы, указана важность/сложность: субъективная, для сферического проекта промышленной админки возрастом более 1 года
- Техдолг — основа для планирования работ
  - Зависимости
  - Размер проекта
  - Комментарии
  - Возраст проекта
  - Статический анализ
  - Стили
  - Покрытие тестами
  - Фреймворк
  - Дублирование кода
  - Правописание

ОТЧЁТ: ЗАВИСИМОСТИ

- важность: средняя
- сложность исправления: средняя
- Всё есть в package.json! Но нет
  - Плохо с человеко-читаемостью, нет группировки
  - Нет ссылок на документацию и дату выпуска библиотеки
- Старые библиотеки
- Структура связей частей кода — граф зависимостей
- Зависимости, которых нет в package.json. Это самописные велосипеды и URL импорты в index.html. Видны на графе.

ОТЧЁТ: РАЗМЕР ПРОЕКТА

- важность: средняя
- сложность исправления: средняя
- Он влияет на всё
- Его необходимо непрерывно сдувать
- Считаем количество строк и файлов

ОТЧЁТ: СТАТИЧЕСКИЙ АНАЛИЗ

- важность: высокая
- сложность исправления: высокая
- Большинство ошибок про стиль кода
- Большая длинна и сложность - самый полезный тип ошибок
- Покрытие типами данных - легче всего исправить, не требует тестирования.
- Асинхронный код - наиболее сложный тип ошибок

ОТЧЁТ: ПОКРЫТИЕ ТЕСТАМИ

- важность: высокая
- сложность исправления: высокая
- Самое неприятное для устранения дело
- Тесты, как правило, тощие
- Можно оценить покрытие скриптами по количеству строк, независимо от фреймворков

ОТЧЁТ: ДУБЛИРОВАНИЕ

- Дублирование кода
- важность: низкая
- сложность исправления: средняя
- проблема "исторически сложилось": некогда думать, надо фигачить в прод
- Битва DRY и WET подходов: баланс между скоростью и качеством

ЧТО ДАЛЬШЕ

- Типичный сценарий:
  - берём задачки техдолга из бэклога, если нечего делать
  - переписываем код, связанный со старыми библиотеками
  - увеличиваем покрытие тестами
  - правим имена переменных/классов/методов
- Прогрессивный сценарий:
  - регулярный анализ состояния техдолга: тонем или всплываем
  - регулярный бюджет для техдолга:
    - один день в спринт
    - спринт через два спринта
    - один спринт после закрытия вехи
  - план работ(вехи) на основе важности, сложности исправления, бюджетов

ПИШИТЕ ХОРОШИЙ КОД!

- Мой телеграм: https://t.me/stepanovv_ru

- Моё сайт: https://stepanovv.ru

- Фоновая музыка: https://www.youtube.com/watch?v=93gZIDZBdtg