Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Кафедра информационных систем и программирования

ОТЧЁТ  
о выполненной лабораторной работе №6

по дисциплине «Разработка, анализ и управление программными проектами»

«ИСПОЛЬЗОВАНИЕ СИСТЕМ КОНТРОЛЯ ВЕРСИЙ ИСХОДНОГО КОДА ПРОГРАММ»

Выполнили:

студенты 4 курса

группы 17-КБ-ПР1

Желябин Григорий

Радуль Иван

Чумаков Илья

Краснодар

2020

**1 Цель работы**

Создать в системе контроля версий репозиторий для нового проекта и выполнить все основные действия с исходным кодом программы, связанные с контролем версий.

**2 Постановка задачи**

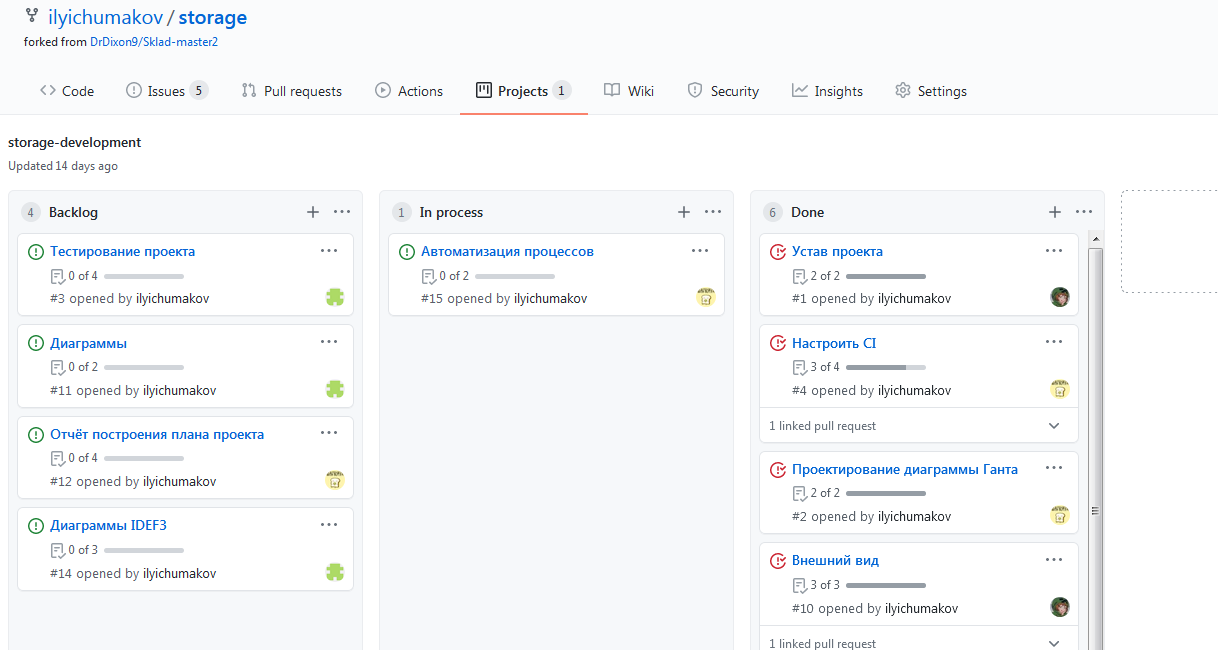
Разработка проектов, для которых требуется более нескольких часов, то есть почти любых, требует использования системы контроля версий. Командная разработка обязывает использовать данный инструмент, иначе неизбежны конфликты изменений.

**3 Задание**

Добавить проект в распределённую систему контроля версий, организовать к нему доступы участников команды. Использовать ветки для независимых изменений кода и последующего их слияния.

**4 Выполнение**

В качестве системы контроля версий был выбран git – современный, простой и надежный инструмент версионирования кода. Для создания нового репозитория испольуется команда git init. Изменённые файлы добавляются в индекс командой git add <path>, фиксируются командой git commit -m "message". Также был создан удаленный репозиторий на портале github.com (<https://github.com/ilyichumakov/storage/>). Там был организован проект, в котором можно управлять задачами, их статусами, тем самым автоматизируя процесс разработки. На рисунке 1 изображена доска Канбан из 3 этапов – Backlog (в очереди на выполнение), In process (в разработке) и Done (выполнено).

Рисунок 1 – Канбан доска проекта в github

Рассмотрим процесс выполнения задачи. Когда разработчик приступает к выполнению, он переносит карточку задачи в соответствующую колонку, создаёт новую ветку с правилом именования: номер задачи, дефис, короткое описание изменений. Например 9-validators.

После проведения нужных правок, изменения фиксируются, отправляются на портал (удаленный сервер), там готовится Pull Request – запрос на внесение изменений. Рекомендуется отключать прямое добавление изменений в основную ветку master в настройках репозитория. Это уберегает от внезапных и нежелательных изменений. На странице создания Pull Request’a вносится заголовок, описываются изменения, есть возможность просмотра изменений относительно целевой ветки, как показано на рисунке 2.

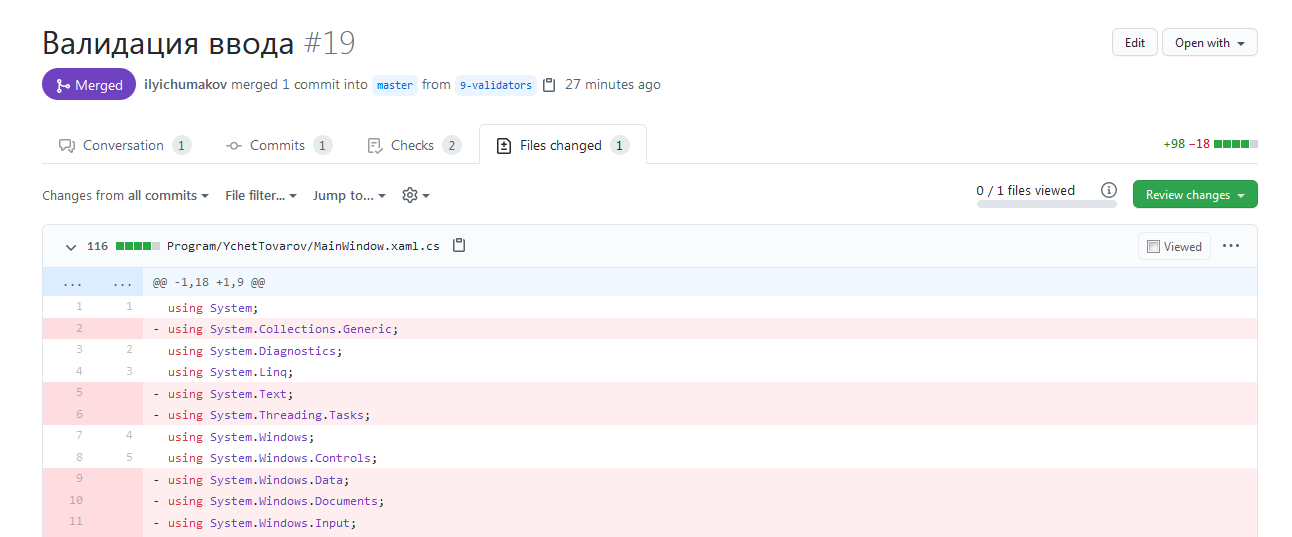


Рисунок 2 – Просмотр вносимых изменений

Это очень удобно использовать для предотвращения добавления в проект «мусорного кода», который мог случайно быть зафиксирован другим разработчиком. Также огромную пользу несёт функционал код ревью. В нашем проекте был установлен запрет слияния изменений без одобрения другим участником команды. Такой подход является хорошей практикой по отлову самых явных ошибок и написанию поддерживаемого кода. Чтобы пометить Pull Request как одобренный, нужно нажать кнопку Review Changes и выбрать пункт Approve, как показано на рисунке 3.

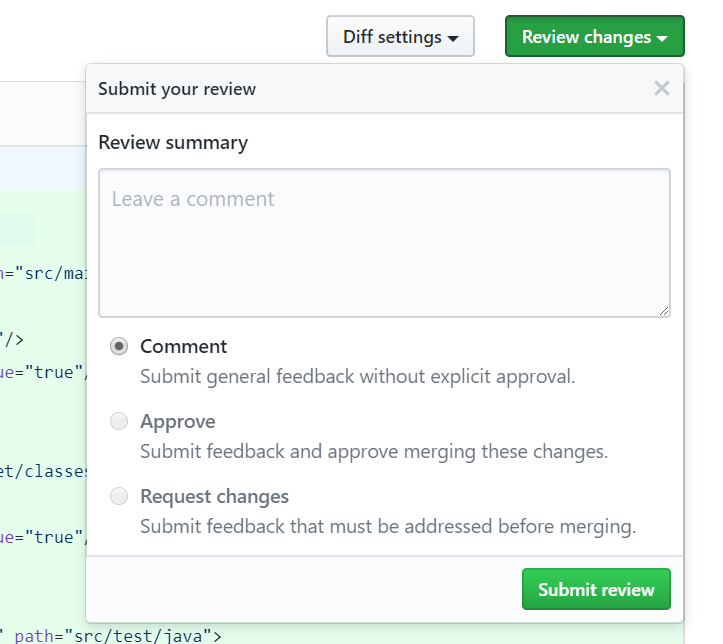


Рисунок 3 – Одобрение изменений другого программиста

Также в нашем проекте был настроен функционал CI (continuous integration) – непрерывная интеграция. Был составлен файл инструкций по проверке работоспособности кода, который получится в результате изменений. В нём происходит попытка сборки решения и запуск тестов. Таким образом независимо от масштаба проекта можно значительно сократить шанс публикации кода с ошибками или вовсе неработоспособного. В Pull Request можно отслеживать статус проверок и запретить слияние при непройденных этапах. Список всех запущенных пайплайнов доступен во вкладке Actions (рисунок 4).

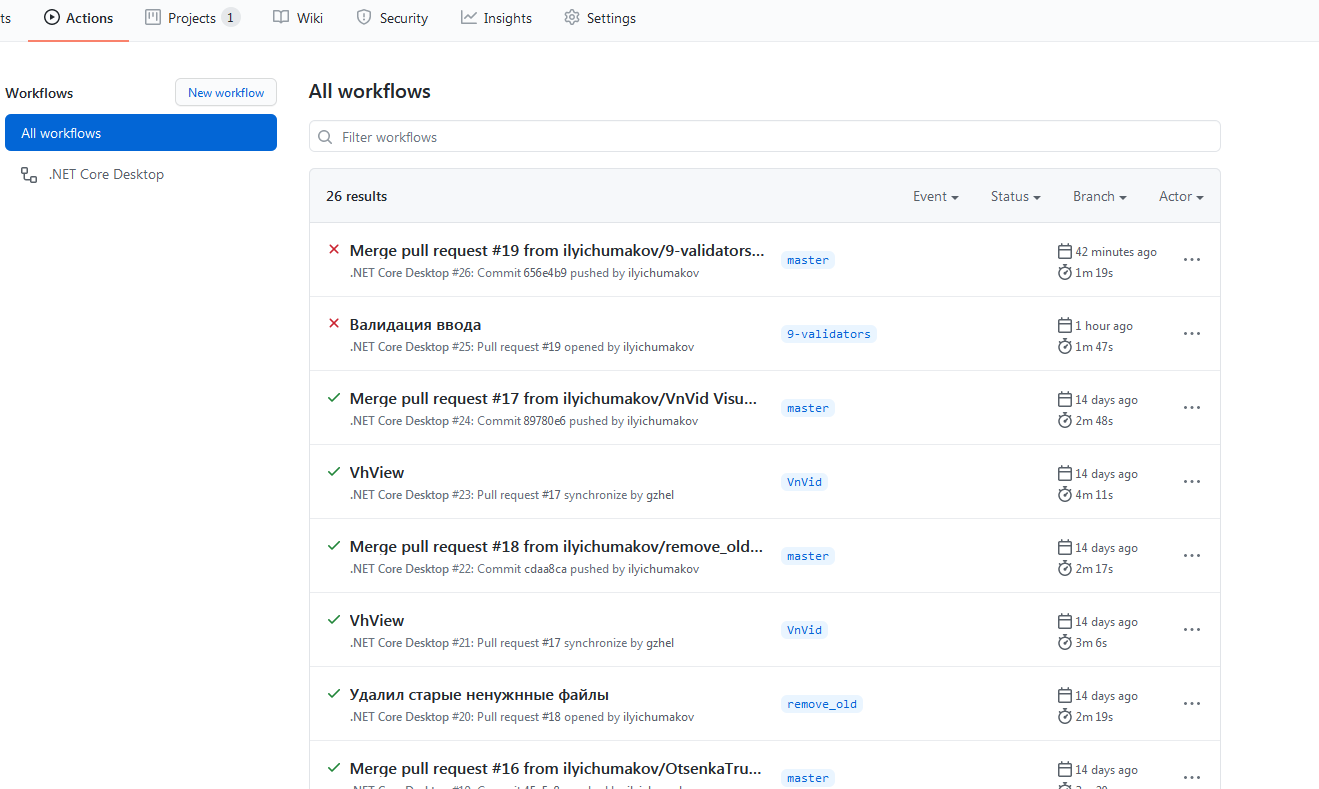


Рисунок 4 – История запуска проверок интеграции