**Отчёт**

**Дисциплина: «Архитектура информационных систем»**

**Лабораторная работа №3 – «Контроль версий, Git, CI/CD»**

Работу выполнил:

Велиев Саидага Саидризаевич

Работу проверил:

Буксиков Артём Аркадьевич

Норильск, 2025г.

**Цель работы**

Освоение ключевых принципов использования системы управления версиями Git, погружение в возможности платформы MosHub и реализация начальной практики CI/CD на примере базового проекта.

**Ход Работы**

В процессе выполнения лабораторной работы я зарегистрировался на платформе MosHub, используя данные учетной записи с портала mos.ru. После авторизации я создал новый приватный проект в личном пространстве, не применяя шаблонов или дополнительных файлов. Полученный URL репозитория будет использоваться для настройки CI/CD и размещения исходных данных лабораторной работы.

Сначала я авторизовался на платформе MosHub, после чего создал пустой приватный репозиторий в своем личном пространстве, не используя шаблоны и дополнительные файлы.

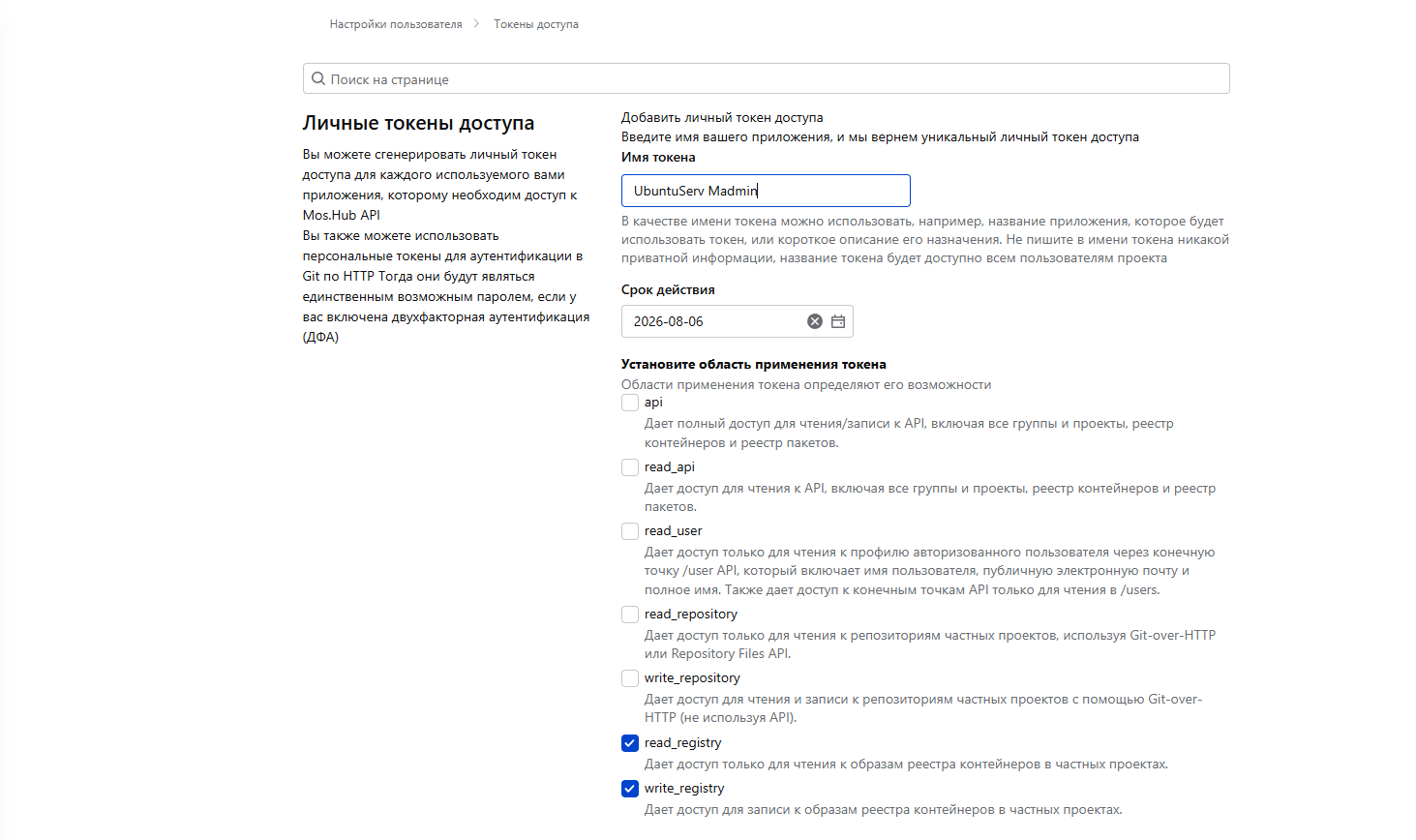


Рисунок 1. Создание пустого репозитория.

Произведена настройка Git в локальной системе. Склонирован удалённый репозиторий на локальную машину.

**

Рисунок 2. Настройка Git и клонирование репозитория.

Исходные файлы лабораторной работы добавлены в репозиторий, зафиксированы первые изменения с комментарием.

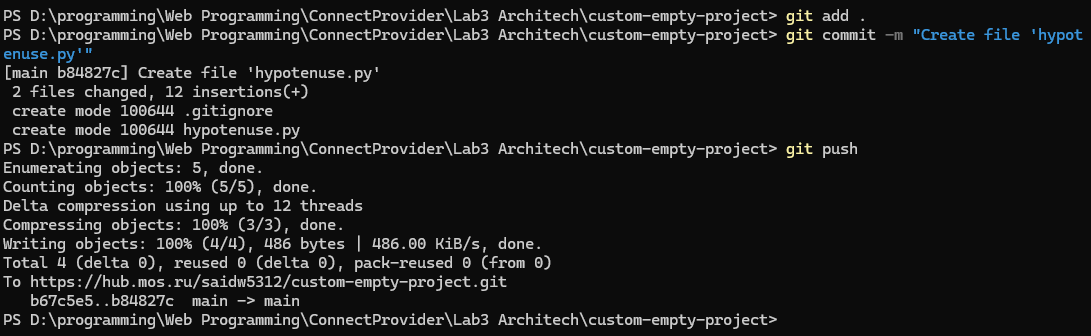
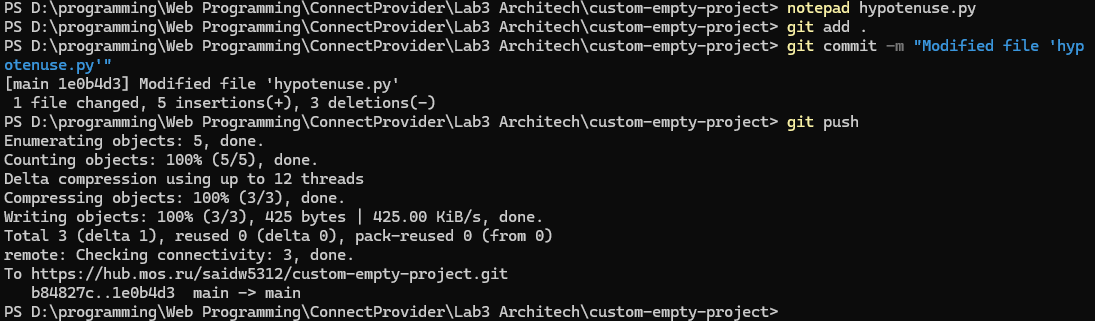


Рисунок 3. Добавление файлов и первый коммит.



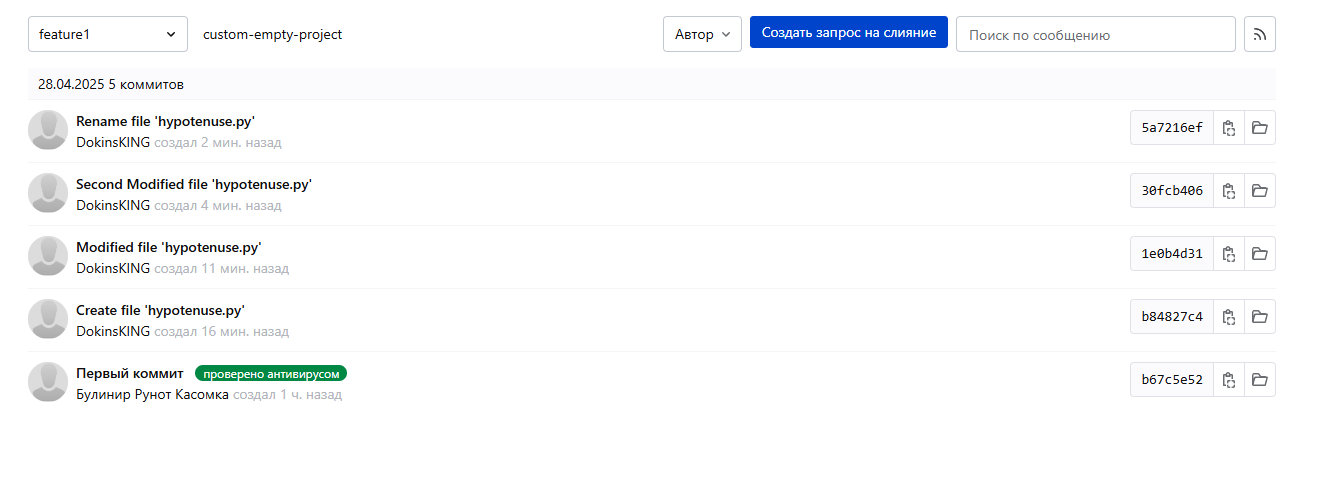
Рисунок 4. Добавление в репозиторий изменений с файлом.

Рисунок 5. История коммитов.

В ходе лабораторной работы были выполнены следующие шаги: создана новая ветка с именем «feature1», в которую были добавлены изменённый файл hypotenuse.py и новый файл calculator.py. После внесения изменений было выполнено слияние веток. Затем был произведен переход на основную ветку main, где изменения были зафиксированы и репозиторий обновлён. Подробности реализации указанных шагов представлены на скриншотах ниже.

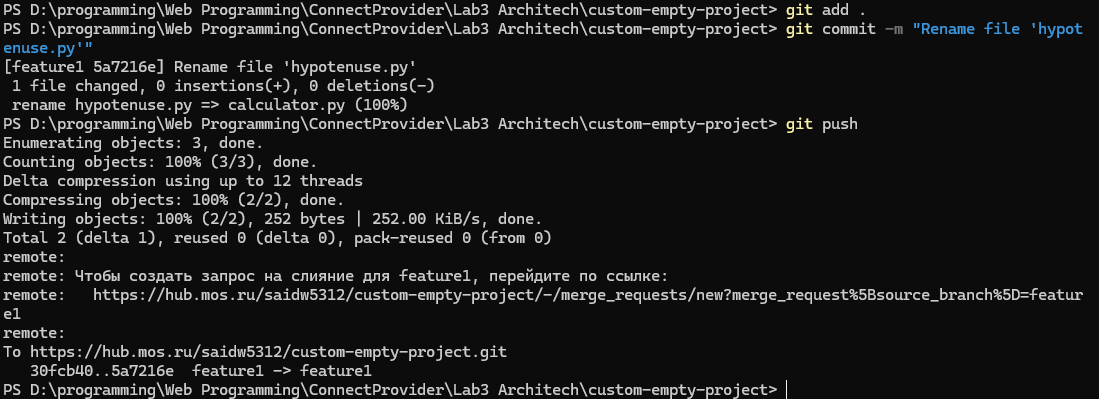


Рисунок 6. Создание ветки и добавление в неё изменений.

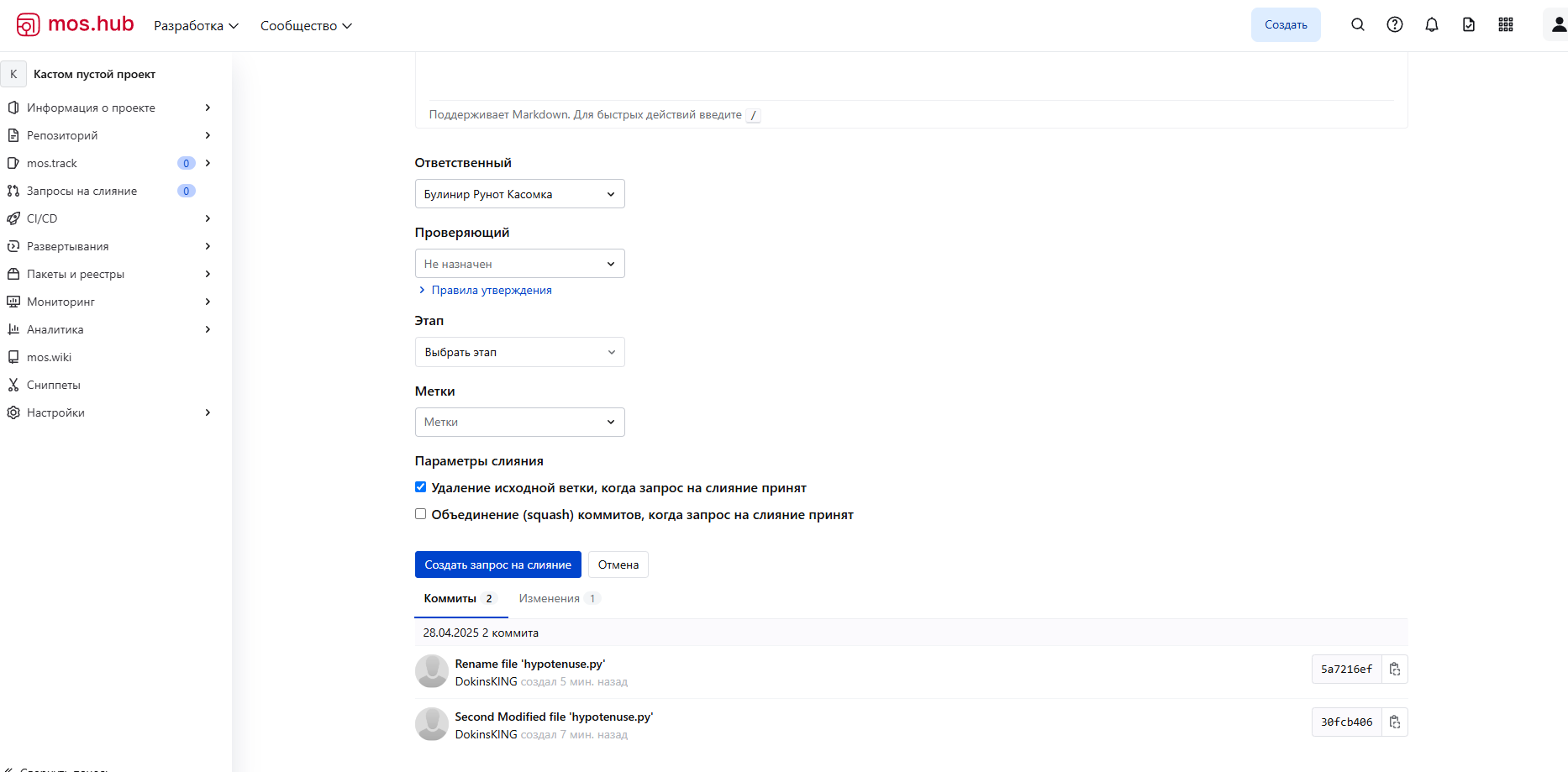


Рисунок 7. Процесс слияния веток.

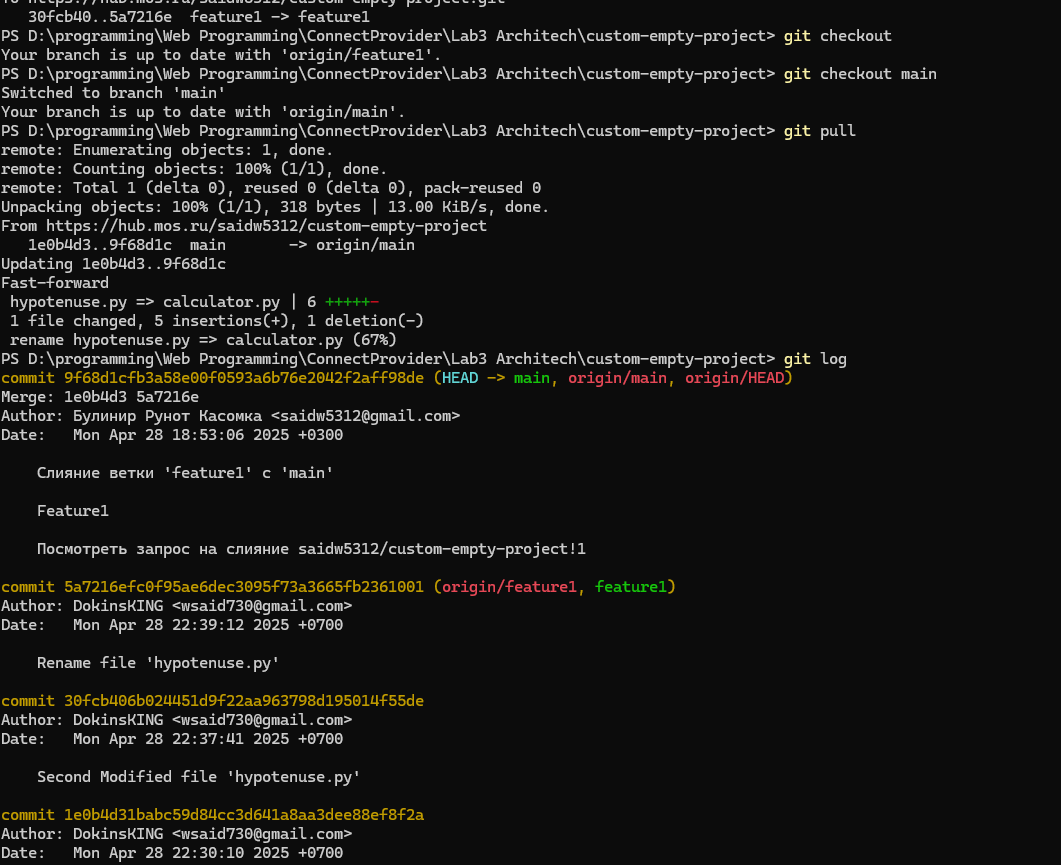


Рисунок 8. Скачивание репозитория после слиянии.

Затем были выполнены следующие действия: создана ветка с именем «patch1», в которой были внесены изменения в файлы проекта. При слиянии этой ветки с основной веткой main был намеренно создан конфликт, который был успешно выявлен и разрешён вручную. После устранения конфликта слияние прошло без ошибок. Подробности выполнения этих шагов можно увидеть на представленных скриншотах.

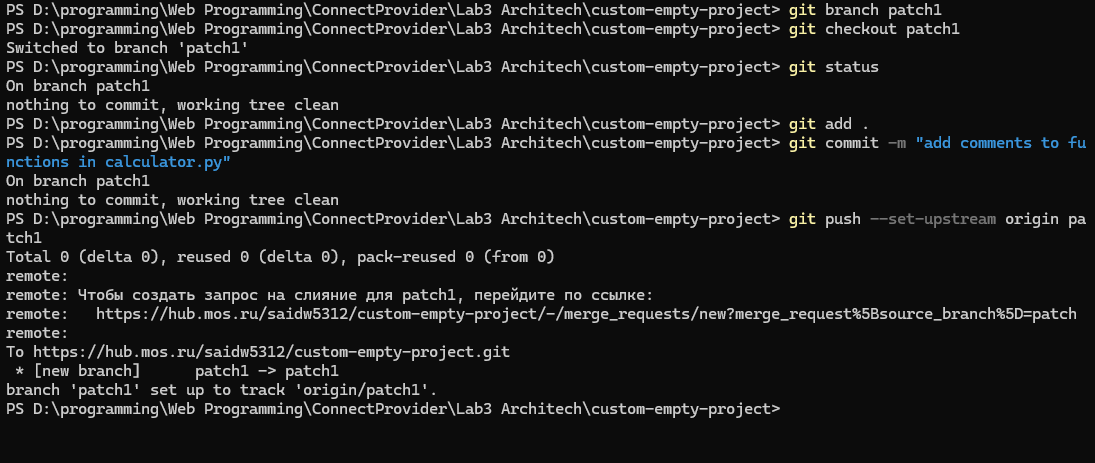


Рисунок 9. Добавление в файл изменений и добавление их в репозиторий.

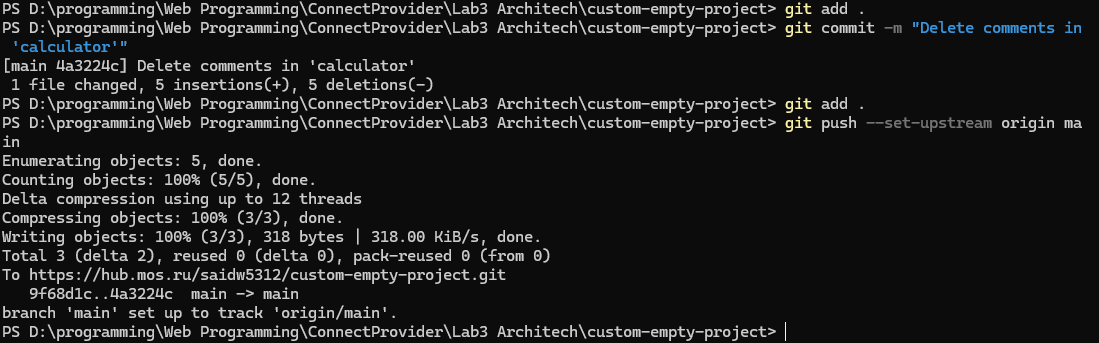


Рисунок 10. Процесс добавления изменений в репозиторий.

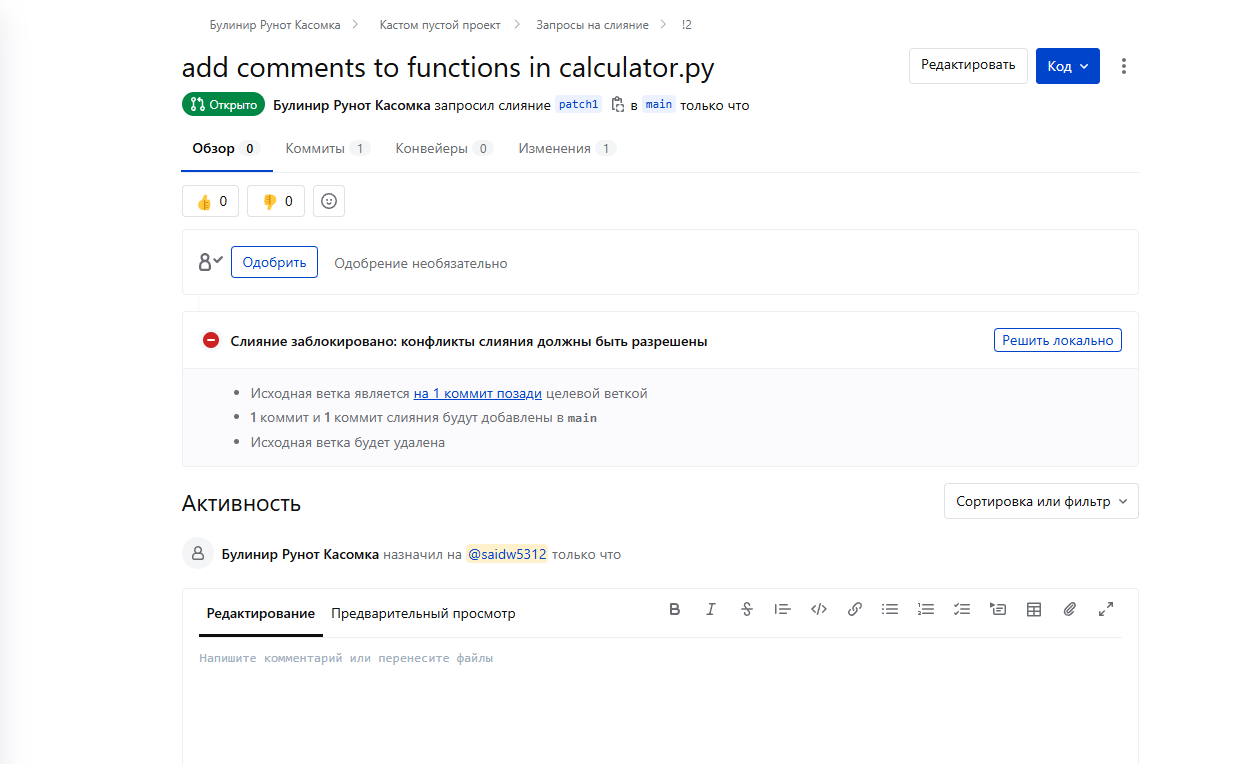


Рисунок 11. Обнаружение конфликта.

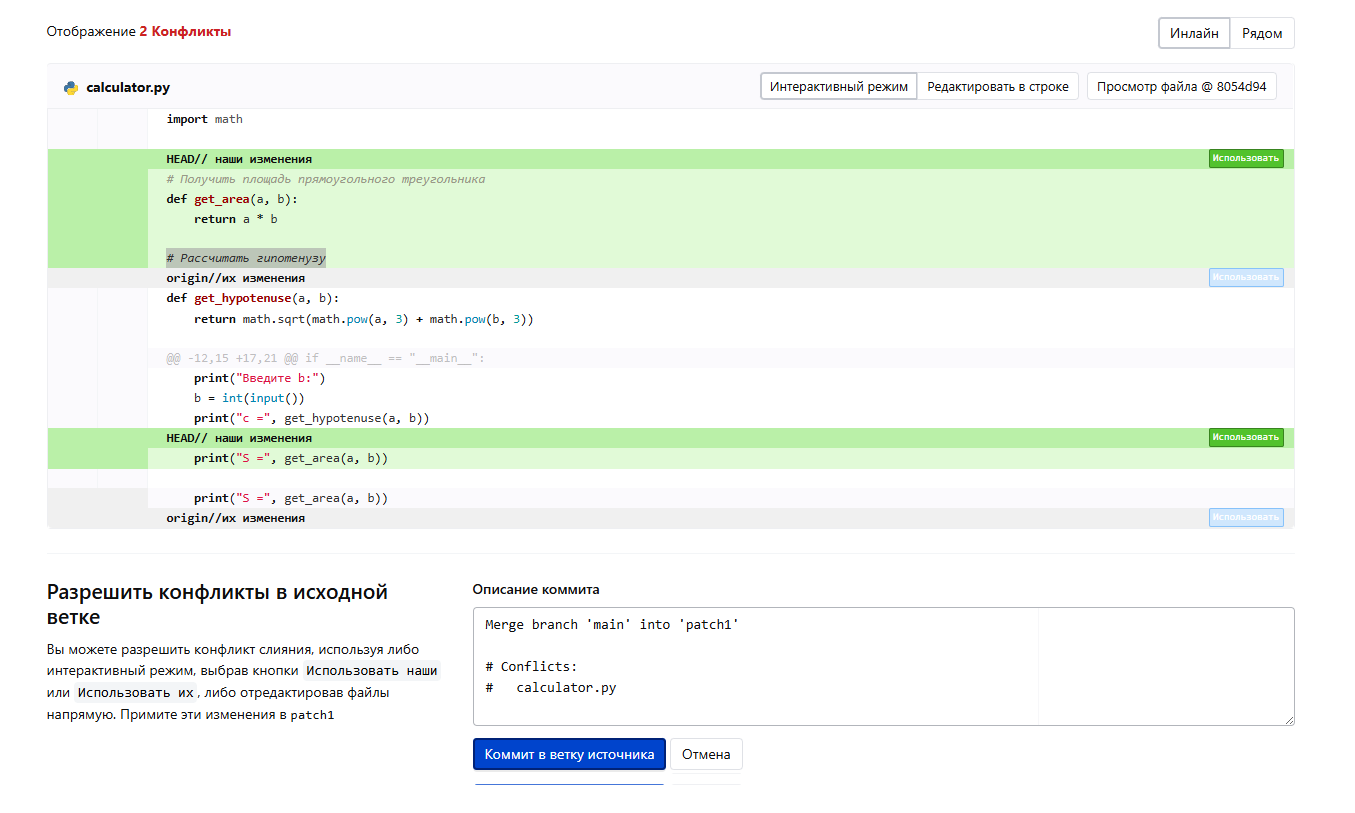


Рисунок 12. Решение конфликта веток.

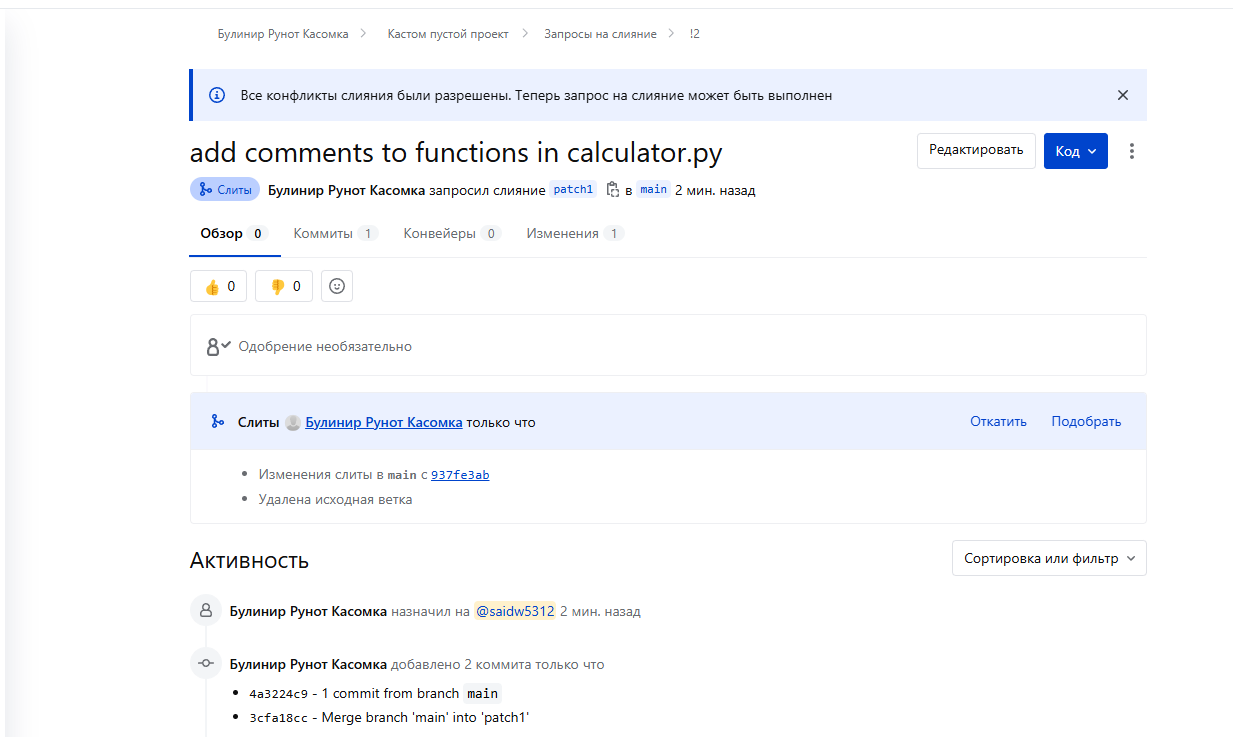


Рисунок 13. Успешное слияние веток.

В процессе выполнения лабораторной работы был развернут GitLab-сервер с помощью Docker-контейнера, настроен gitlab-runner для автоматического выполнения тестов, а также создан конфигурационный файл .gitlab-ci.yml, определяющий этапы CI-процесса. После завершения настройки были проведены тесты для проверки правильности работы проекта. Скриншоты, на которых отображены все шаги, приведены ниже.

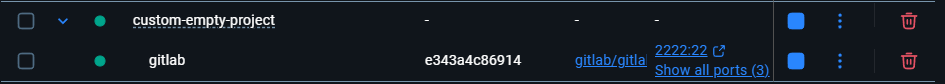


Рисунок 14. Развертывание образа GitLab в докере.

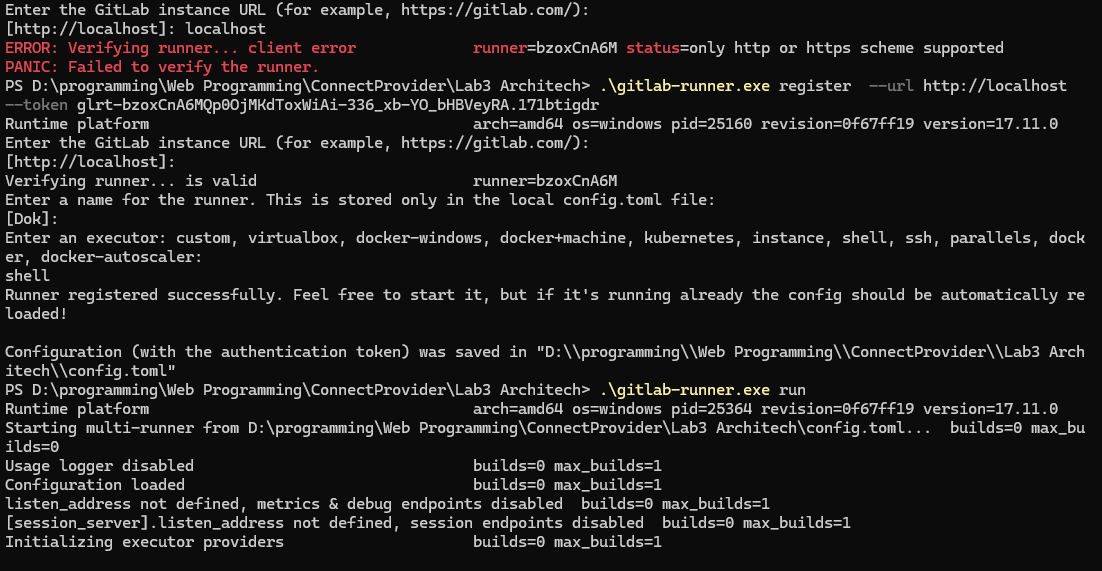


Рисунок 15. Настройка gitlab-runner.

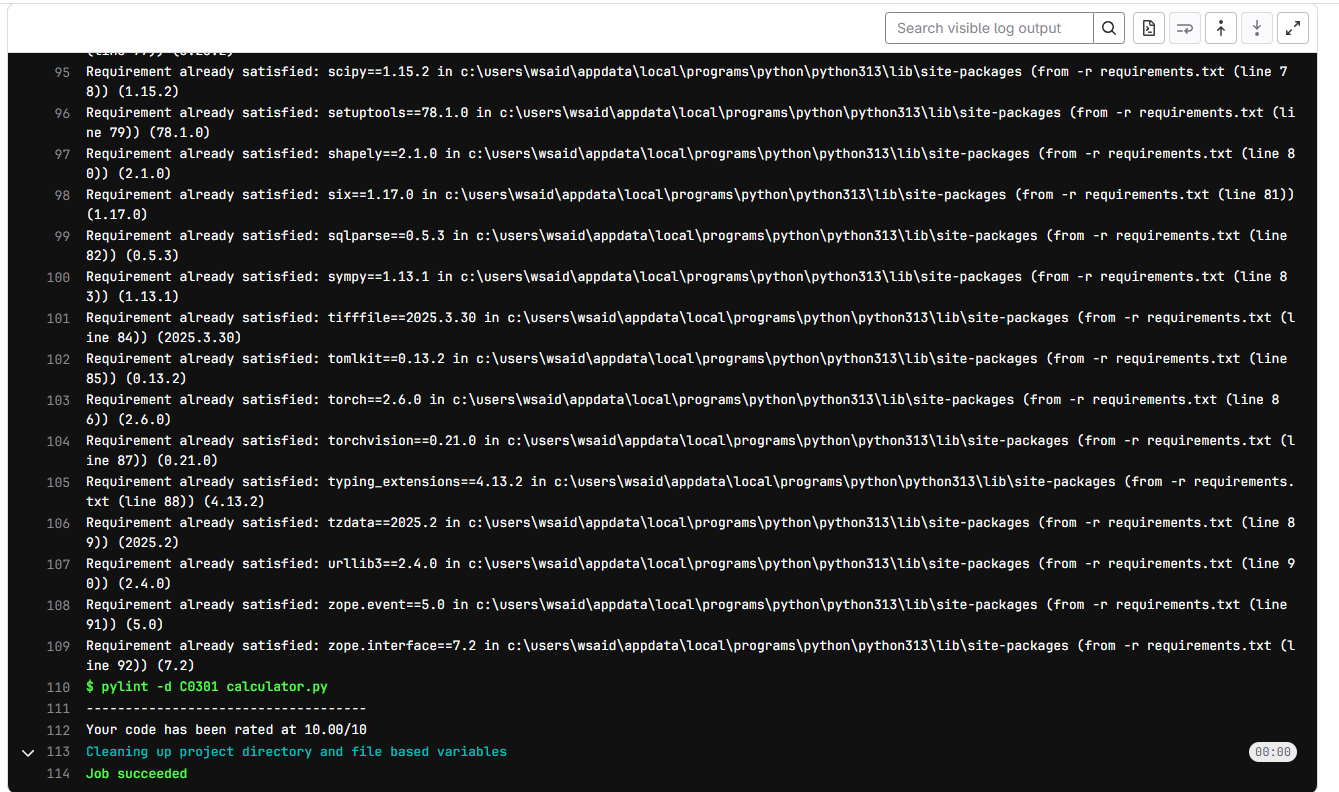


Рисунок 16. Проведение тестов.

**Заключение**

В результате выполнения лабораторной работы были освоены базовые и продвинутые приёмы работы с системой контроля версий Git, а также выполнена интеграция с отечественной платформой MosHub. В рамках задания был создан и наполнен репозиторий, настроен CI/CD пайплайн для автоматической проверки проекта при каждом коммите. Дополнительно проведена настройка автотестирования в GitLab, включая развёртывание сервиса через Docker и конфигурацию gitlab-runner. Изученные технологии и выполненные практические действия формируют прочную основу для последующей автоматизации процессов разработки, тестирования и развёртывания программных продуктов.