МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности

Кафедра компьютерных интеллектуальных технологий проектирования

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

По дисциплине: «Системы хранения и обработки данных»

Тема: «Интеграция изменений в системе управления версиями Git»

Выполнил работу студент группы змИИВТ-241: Фролов Д.С.

подпись, дата

Принял: Короленко В.В.

подпись, дата

Воронеж 2024

Лабораторная работа №5

«Интеграция изменений в системе управления версиями Git»

1. Цель лабораторной работы: изучить процесс интеграции изменений в системе управления версиями с помощью операции Pull Request (Merge Request).

2. Задачи лабораторной работы:

- перенос (интеграция) изменений в главную ветку с помощью операции pull request (merge request);

- рецензирование проекта коллегами и рецензирование проектов коллег.

3. Результаты лабораторной работы

Произведем интеграцию проекта, т.е. перенесем всю накопленную информацию в главную ветку (main). Для того чтобы перенести файлы проекта из ветки LR4-FDS в ветку main, необходимо создать и провести операцию pull request (merge request). Для этого на вкладке «Pull requests» нажмем на кнопку «New pull request». В качестве основной ветки выберем ветку «main», а в качестве вливаемой ветки – ветку «LR4-FDS» (рисунок 1).

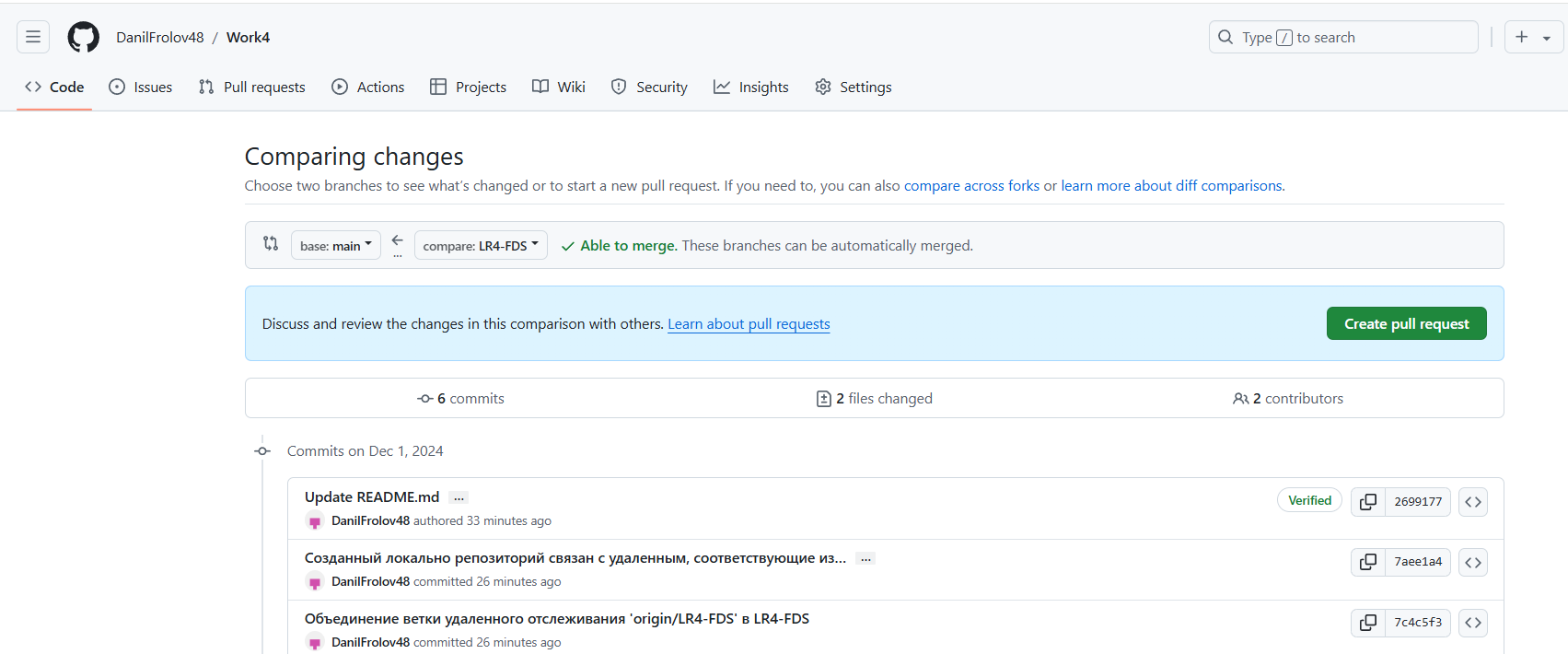


Рисунок 1 – Создание слияния через интерфейс GitHub

Далее нажимаем на зеленую кнопку «Create pull request» и заполняем поля «Add a title» и «Add a description» нашего pull request (рисунок 2).

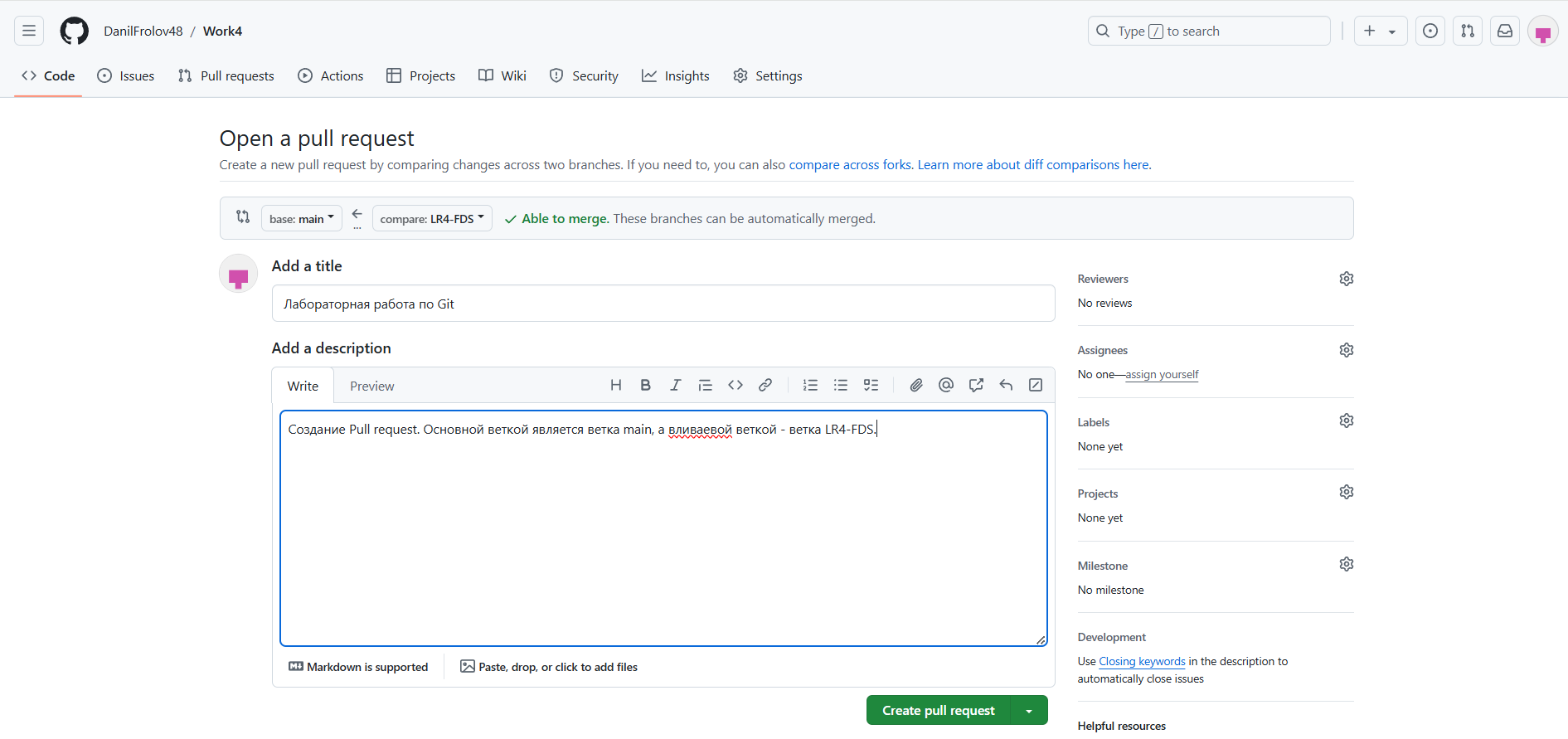


Рисунок 2 – Вкладка «Open a pull request»

Чтобы была возможность добавить ревьюера, этот пользователь должен быть включен в список соучастников проекта (collaborators). Для этого в настройках выбираем соответствующую вкладку и нажимаем на кнопку «Add people». Вводим имя пользователя и добавляем в список «Collaborator». Результат представлен на рисунке 3.

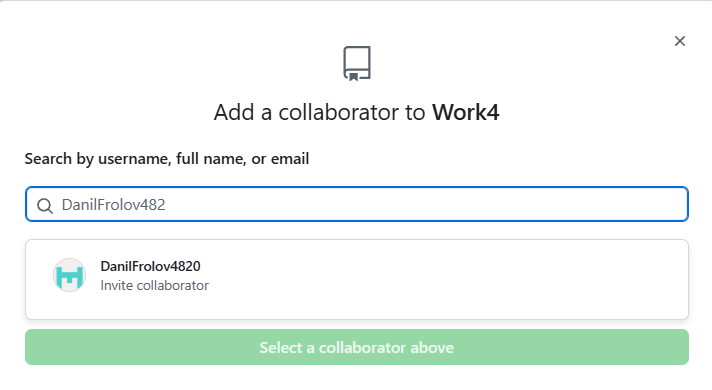


Рисунок 3 – Добавление пользователя в список соучастников проекта

После этого пользователь должен принять приглашение, и он будет доступен в списке ревьеров. Результат представлен на рисунках 4-5.

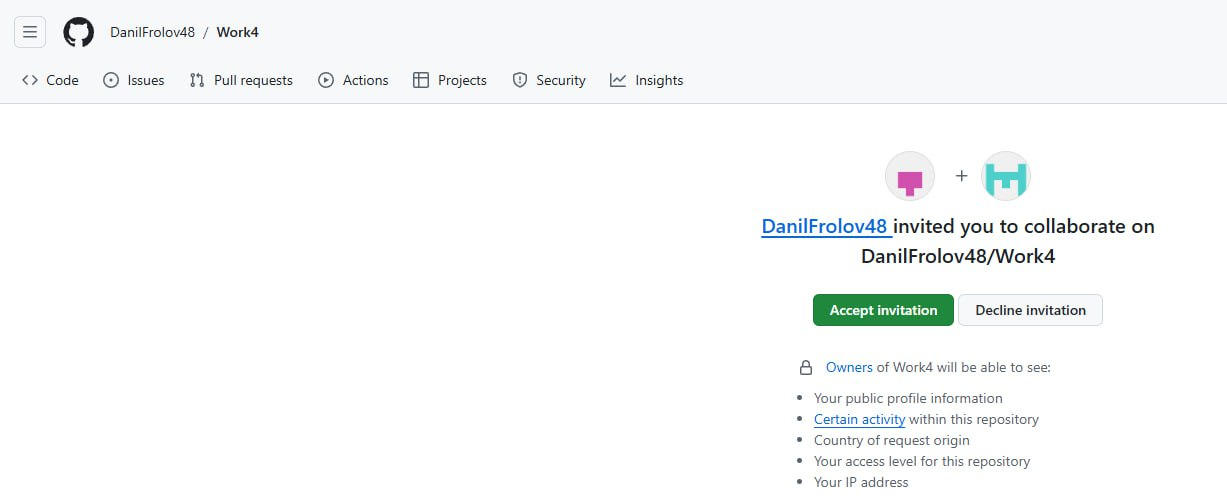


Рисунок 4 – Уведомление пользователя DanilFrolov4820 присоединиться в проект

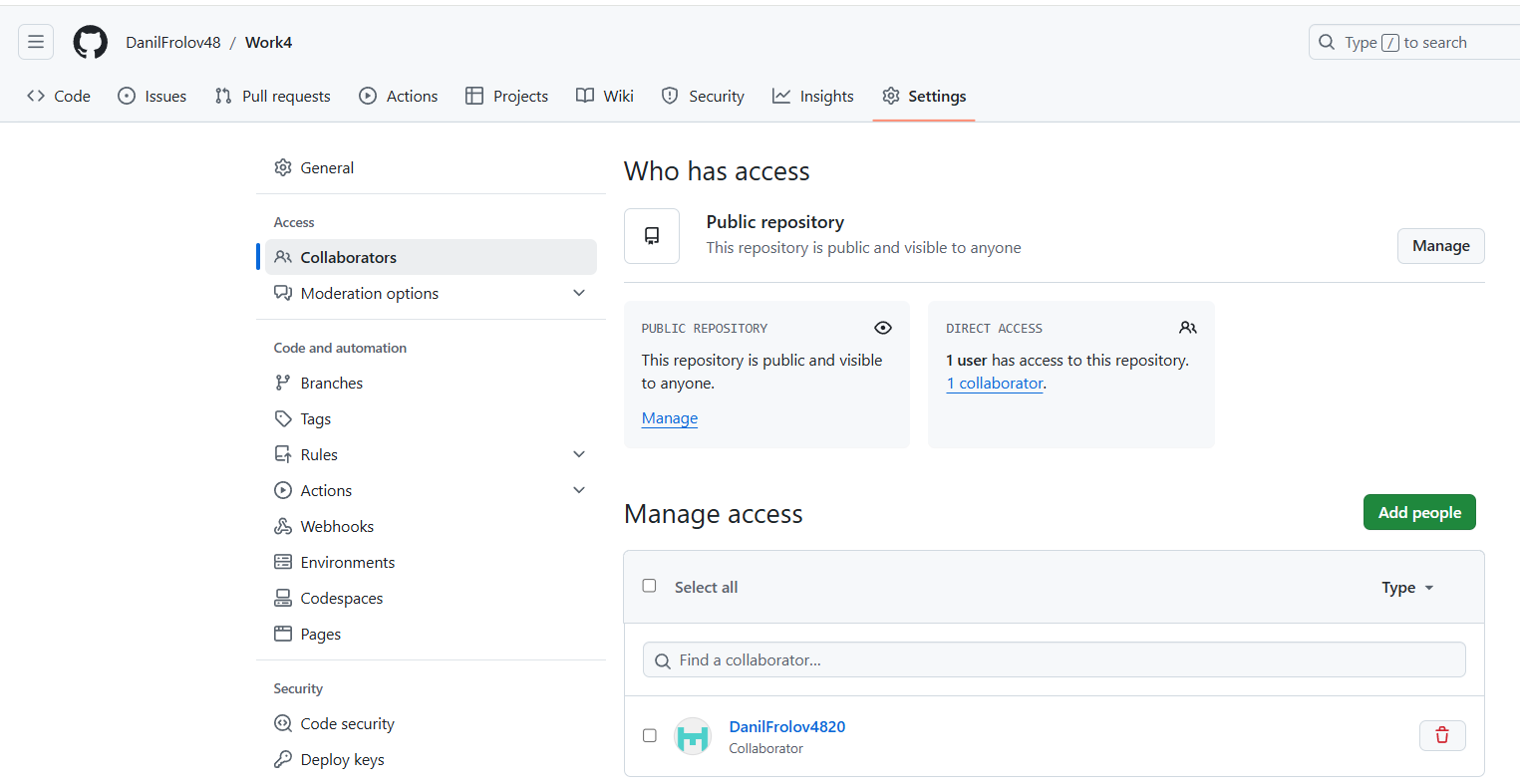


Рисунок 5 – Успешное добавление пользователя в проект

Далее необходимо сделать так, чтобы перед выполнением Pull request требовалось одобрение другого пользователя. Для этого пользователю DanilFrolov48 необходимо в верхней части страницы нажать на вкладку «Settings». Далее открыть раздел «Branches». В разделе «Branch protection rules» нажать на кнопку «Add classic branch protection rule» (рисунок 6).

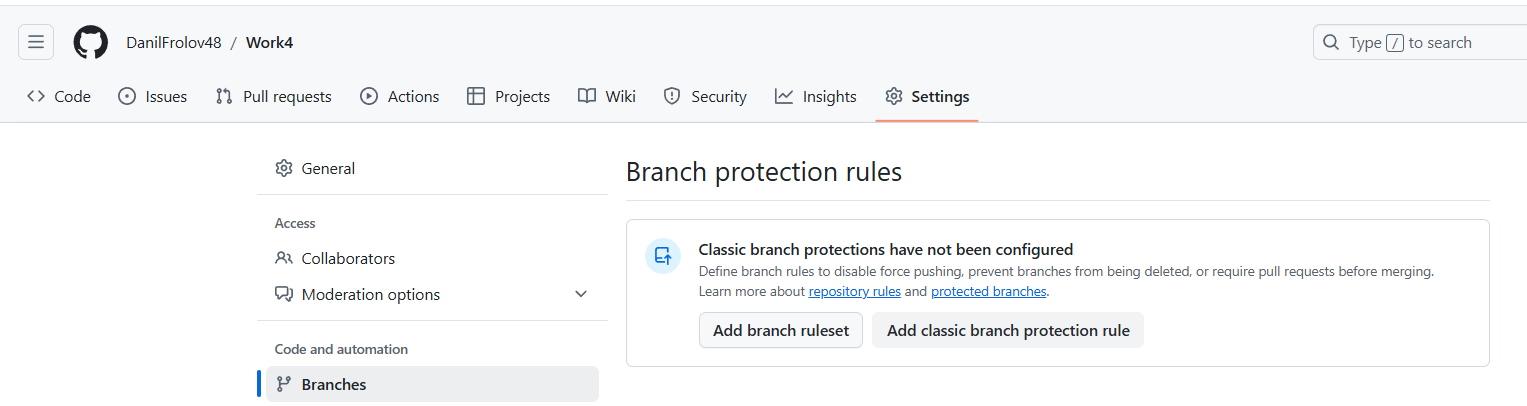


Рисунок 6 – Раздел «Branch protection rules»

В поле Branch name pattern введем «main», чтобы применить правило к основной ветке «main». Поставим галочку напротив опции «Require a pull request before merging». После этого появятся дополнительные настройки. Активируем опцию «Require approvals». Результат выполнения данных действий представлен на рисунке 7.

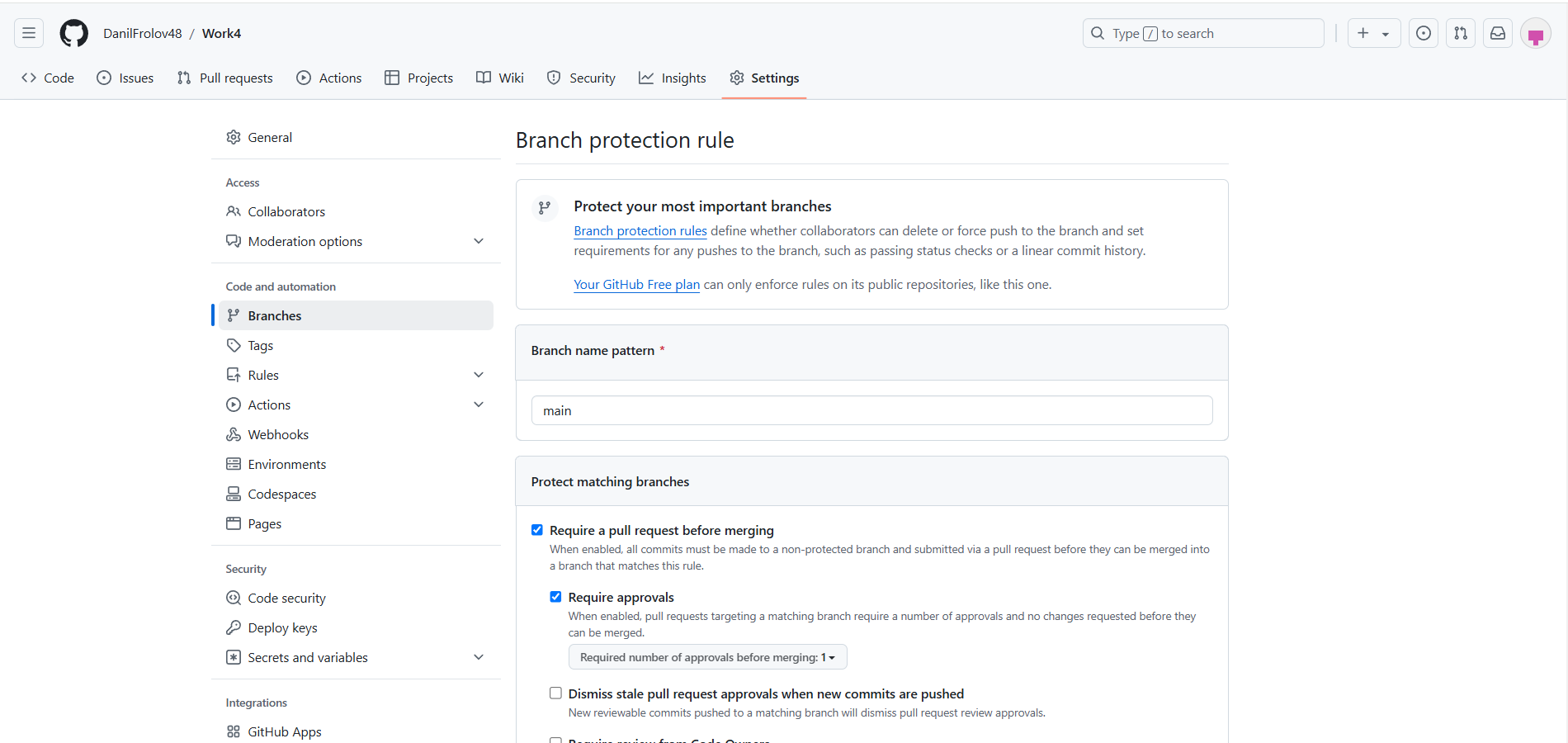


Рисунок 7 – Раздел «Branch protection rule»

Внизу страницы раздела «Branch protection rule» нажмем на зеленую кнопку «Create».

Далее добавляем ревьюера в Pull request на GitHub. Для этого выполняем следующие шаги:

1. Открываем страницу pull request в своем репозитории на GitHub;

2. Щелкаем на кнопку «Reviewers» справа от экрана;

3. Вводим имя пользователя ревьюера в поле «Add a reviewer» и выбираем его из списка предложенных пользователей;

4. Нажимаем на кнопку «Add Reviewer» для добавления ревьюера в Pull request.

Результат выполнения данных действий представлен на рисунке 8.

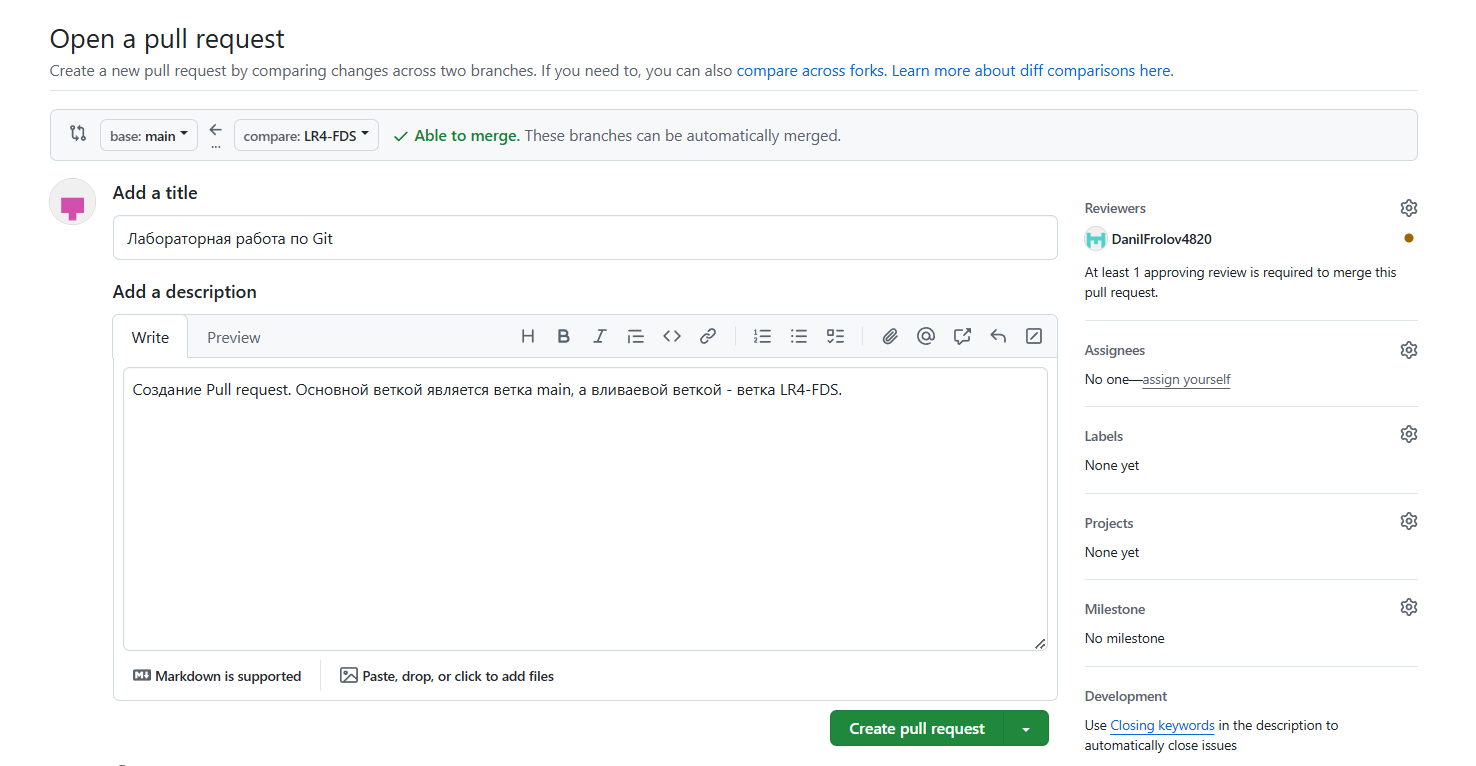


Рисунок 8 – Добавление ревьюера в Pull request

Далее нажимаем на зеленую кнопку «Create pull request». Пользователю DanilFrolov48 запрещено выполнять Pull request до того момента, пока ревьюер DanilFrolov4820 не одобрит проведение Pull Request (рисунок 9).

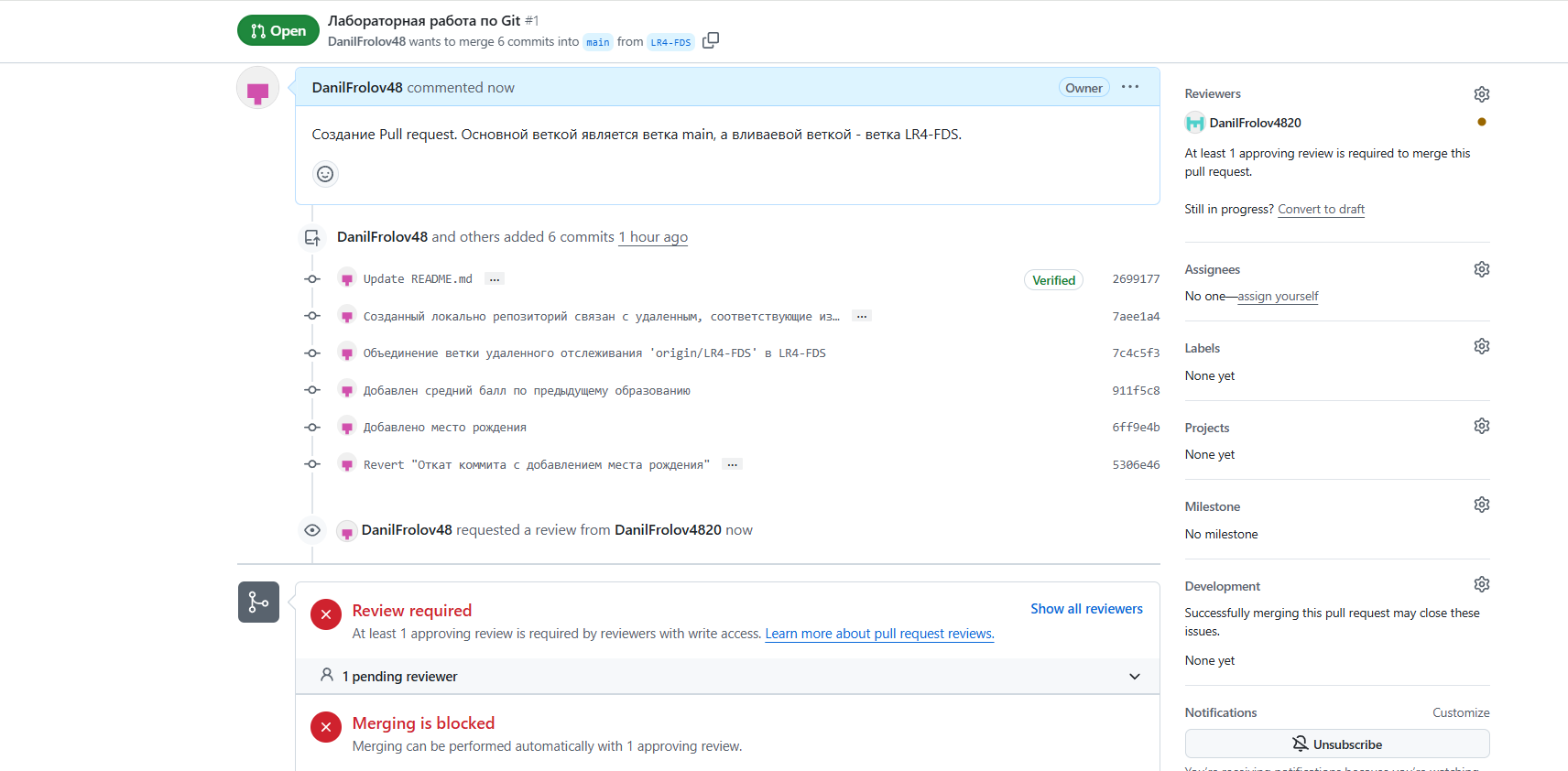


Рисунок 9 – Запрет на проведение Pull request без одобрения ревьюера

Далее ревьюеру DanilFrolov4820 необходимо дать одобрение проекту перед выполнением Pull request. Для этого необходимо выполнить следующие шаги:

1. Перейти в нужный репозиторий на GitHub. В верхней части страницы выбрать вкладку «Pull requests». Найти и открыть Pull request, который необходимо просмотреть и одобрить;

2. В интерфейсе Pull request выбрать вкладку «Files changed». Ознакомиться с внесенными изменениями, просматривая строки кода;

3. Нажать на зеленую кнопку Review changes, расположенную в правом верхнем углу;

4. Оставить комментарий к проекту. Если все изменения устраивают, то выбрать опцию «Approve» и нажать на зеленую кнопку «Submit review» для подтверждения вашего решения.

Результат выполнения данных действий представлен на рисунке 10.

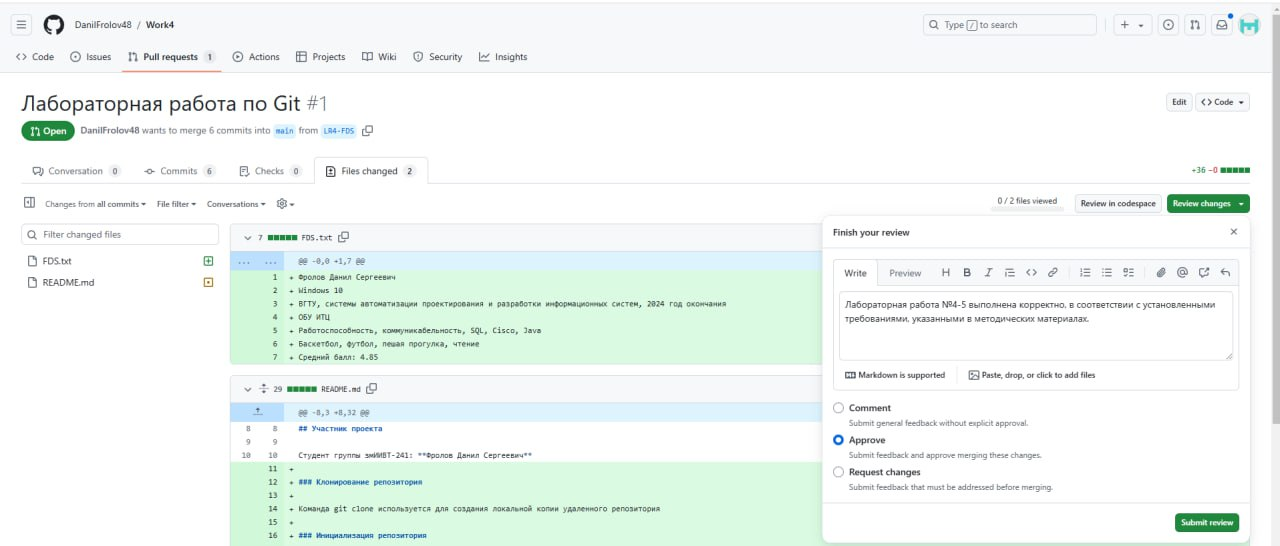


Рисунок 10 – Выдача одобрения проекту перед выполнением Pull request

Теперь пользователь DanilFrolov48 может нажать на зеленую кнопку «Merge pull request» и подтвердить слияние, потому что ревьюер выдал одобрение проекту перед выполнением Pull request (рисунок 11).

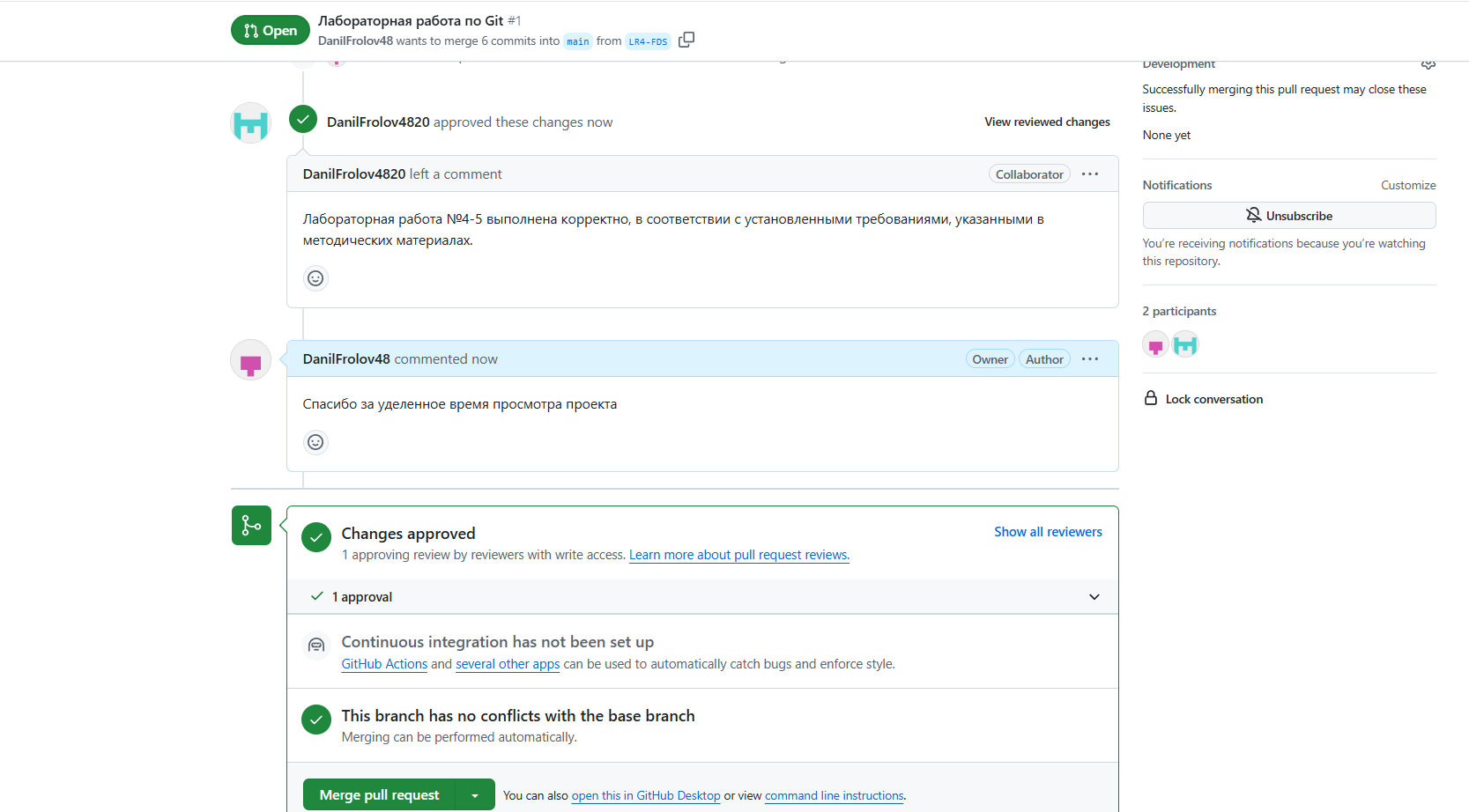


Рисунок 11 – Ревьюер выдал одобрение проекту перед выполнением Pull request

Далее пользователь DanilFrolov48 нажимает на зеленую кнопку «Merge pull request», далее на кнопку «Confirm merge» и удаляет ветку «LR4-FDS». Результат выполнения представлен на рисунке 12.

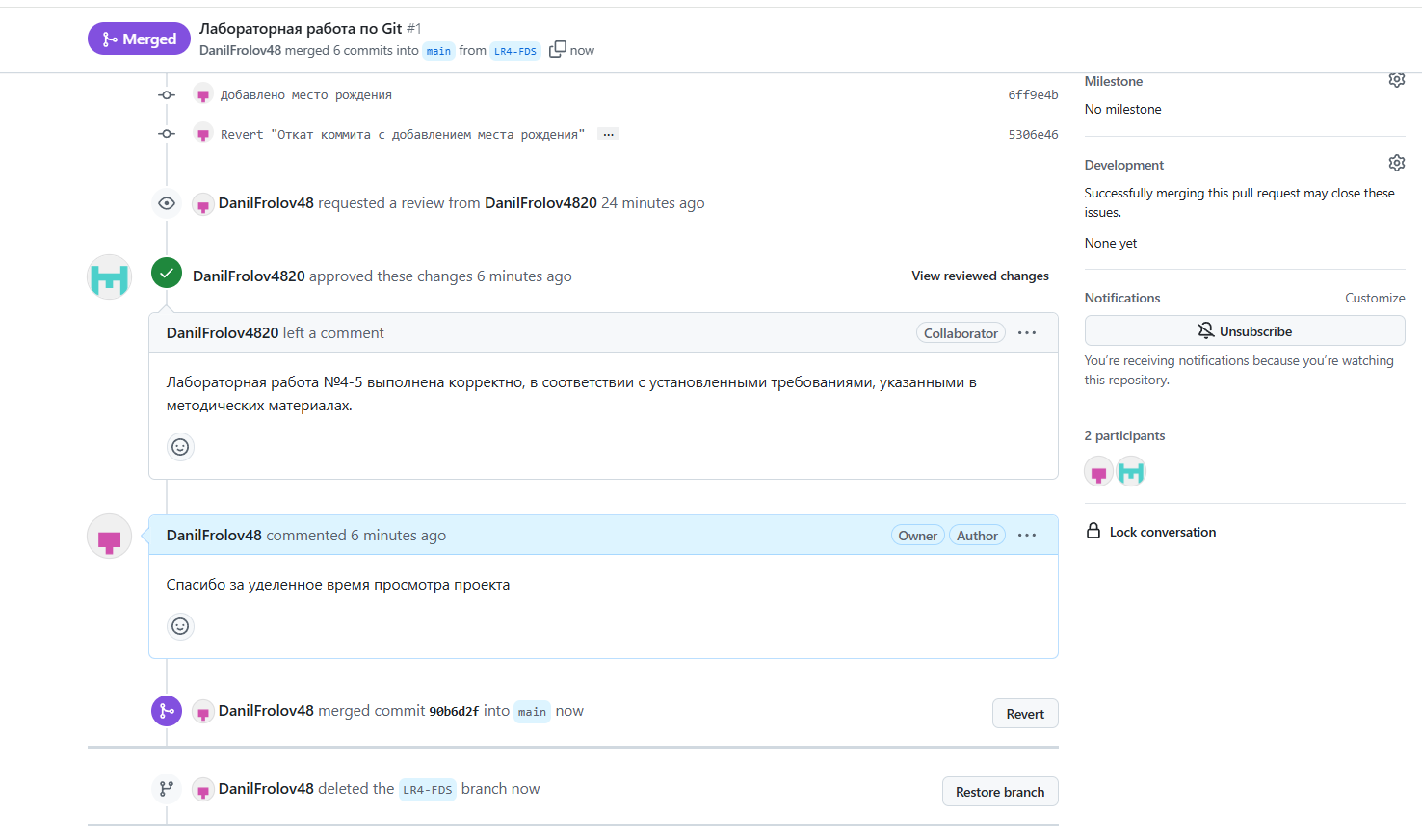


Рисунок 12 – Результат создания слияния

4. Выводы по результатам лабораторной работы

В ходе выполнения данной лабораторной работы был изучен процесс интеграции изменений в системе управления версиями с помощью операции Pull Request (Merge Request).

5. Ответы на контрольные вопросы

1. При работе с системой контроля версий Git можно в любой момент отменить внесенные изменения. В Git существует несколько способов, которые позволяют отменить последние внесенные изменения в необходимом коммите или ветки.

Одним из основных способов отмены (удаления) внесенных изменений в Git является команда git reset. Команда git reset предназначена для удаления коммита путем переноса указателя HEAD назад к более старому (предыдущему) коммиту. Тем самым git reset эмулирует ситуацию, как будто последнего коммита вообще не существовало. Предположим, мы только что сделали коммит, но поняли, что забыли внести изменения или добавить нужный файл. Один из способов отмены внесенных изменений – сделать откат к предыдущему коммиту. Для этого воспользуемся следующей командой: git reset HEAD~1, где:

- HEAD – указатель на текущую ветку, а также указатель на последний коммит в этой ветке;

- ~1 – количество коммитов, на которое необходимо произвести откат. В данном случае откат будет произведен на один коммит назад.

Команда выше произведет откат на один коммит назад, при этом текущая рабочая директория и индекс Git останутся неизменными. Команда git reset поддерживает несколько различных режимов.

- режим soft. Данный режим предназначен для отмены последних коммитов. Изменения, которые были внесены в сами файлы, остаются в индексе. При использовании режима soft указатель HEAD перемещается на выбранное количество коммитов назад. Изменения остаются в разделе проиндексированных файлов и в рабочем каталоге. Дерево объектов также не изменяется;

- режим mixed. Данный режим используется по умолчанию. Предназначен для отмены последних внесенных изменений в коммите, а также сбрасывания индекса. При этом рабочая директория остается без изменений;

- режим hard. Данный режим предназначен для полного удаления всех изменений, которые были сделаны после коммита. Будут удалены любые изменения, внесенные в рабочую директорию и индекс. Отменить действия, выполненные при помощи git reset --hard, нельзя, поэтому необходимо быть осторожными при использовании данного режима. В качестве меры предосторожности рекомендуется сделать резервную копию файлов, к которым будет применяться режим hard.

Еще одним способом отмены внесенных изменений является команда git revert. В то время как git reset изменяет историю репозитория при помощи перемещения указателя ветки (HEAD) на необходимый коммит, git revert создает новый коммит, а также не удаляет историю с ранее внесенными изменениями (коммитами). Тем самым вы полностью сохраняете структуру изменений вашего проекта.

Еще одним способом отмены внесенных изменений является команда git checkout. Несмотря на то, что данная команда предназначена для работы с ветками (в частности, для создания новых веток, а также для переключения между существующими ветками), ее можно использовать для отката изменений в одном определенном файле или в нескольких файлах сразу. Предположим, что мы внесли некоторые изменения в файл script.sh и сделали коммит, однако вскоре поняли, что внесенные изменения нас не устраивают и нам необходимо вернуться к последнему коммиту для файла script.sh. В таком случае необходимо выполнить команду: git checkout HEAD --script.sh. Команда выше вернет состояние файла script.sh к последнему коммиту, который был сделан для файла script.sh. Если же необходимо вернуться не к последнему коммиту, а к другому, то необходимо знать хэш-сумму нужного коммита. Для этого необходимо воспользоваться командой git log, которая отобразит историю коммитов. Хэш-сумма отображается после слова commit, тогда команда будет выглядеть следующим образом: git checkout <хэш-сумма> --script.sh.

2. Публичный репозиторий – это репозиторий, доступ к которому открыт для всех пользователей. Любой человек может просматривать его содержимое, клонировать, делать форки, а в зависимости от настроек прав – вносить изменения (через pull requests). Обычно публичные репозитории используются для проектов с открытым исходным кодом, чтобы разработчики из разных частей мира могли совместно работать над проектом.

3. Pull Request – это операция в системах контроля версий, таких как Git, которая позволяет разработчикам предлагать свои изменения кода для интеграции в основную кодовую базу проекта. Основные цели Pull Request:

- совместная работа. Позволяет нескольким разработчикам работать над разными частями проекта одновременно, объединяя свои изменения в единую кодовую базу;

- код-ревью. Предоставляет возможность другим участникам проекта просмотреть и обсудить внесенные изменения, что способствует улучшению качества кода и выявлению потенциальных ошибок.

Pull Request работает следующим образом:

- создание ветки. Разработчик создает новую ветку в репозитории, где будет вносить изменения. Это позволяет изолировать работу над определенной задачей;

- внесение изменений. Ветка редактируется, добавляются новые функции, исправляются ошибки или вносятся другие улучшения;

- commit и push. Изменения коммитятся (сохраняются) локально и затем отправляются в удаленный репозиторий на платформу GitHub;

- создание pull request. Разработчик инициирует pull request из своей ветки в целевую ветку основного репозитория (обычно это main). В pull request описывается цель изменений, добавляется описание;

- код-ревью. Другие разработчики проекта просматривают изменения, оставляют комментарии, задают вопросы или предлагают улучшения;

- внесение правок. На основании полученных отзывов разработчик может внести дополнительные изменения и обновить Pull Request;

- слияние (merge). После одобрения pull request, изменения интегрируются в целевую ветку;

- закрытие pull request. После слияния pull request закрывается, и ветка может быть удалена, если она больше не нужна.