Московский Авиационный Институт

Институт №3

# «Системы управления, информатика и электроэнергетика»

Кафедра №304.

«Вычислительные машины, системы и сети»



Отчет по практической работе №1

«Система контроля версий Git»

по учебной дисциплине «Технология разработки программного обеспечения»

Вариант №1

Выполнил:

Благушин Н.А.

Группа: М3О-107СВ-24

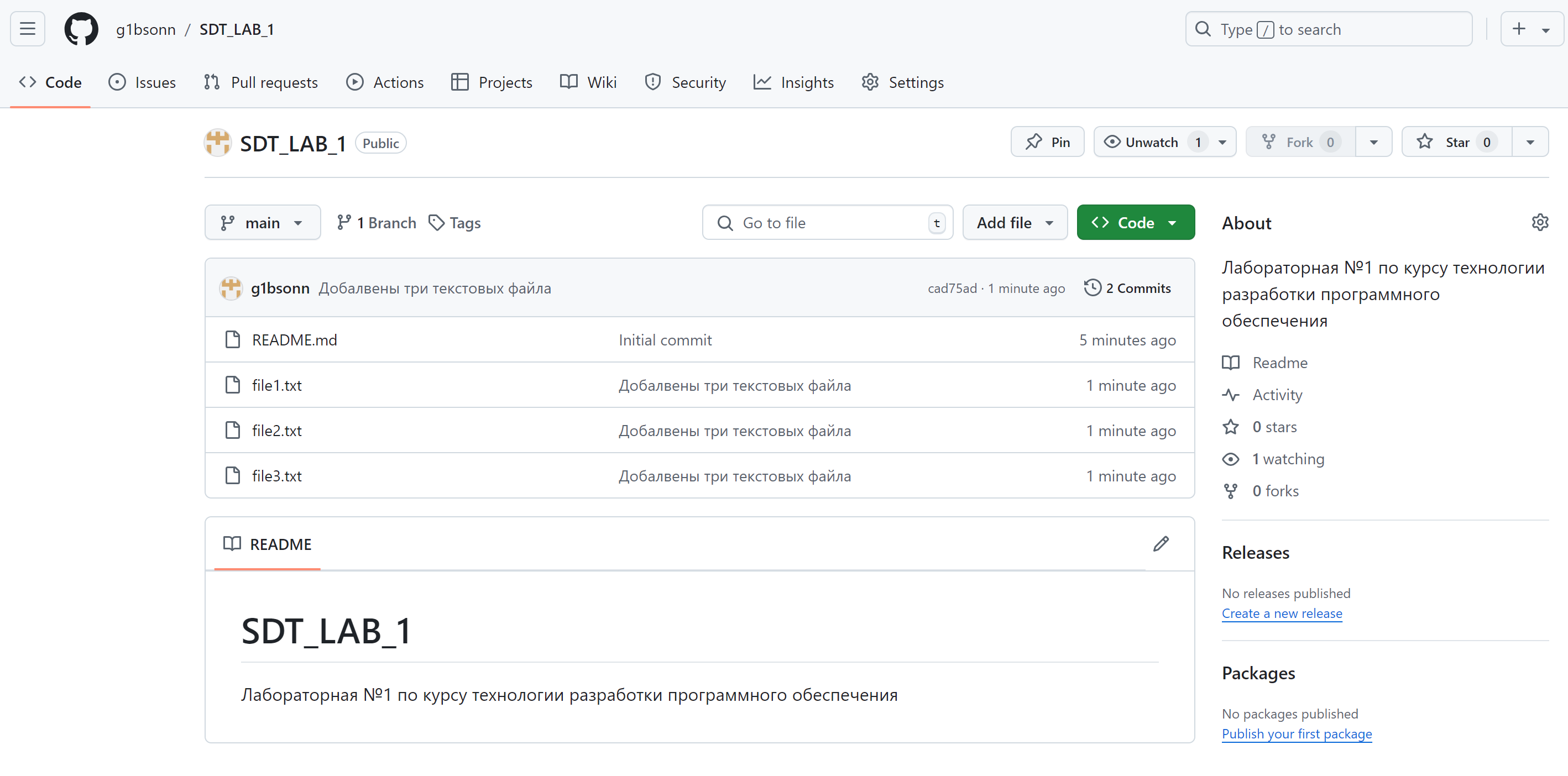
Принял:

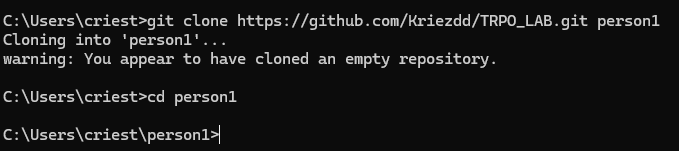
Титов Ю. П.

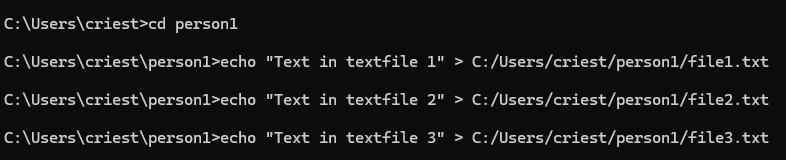
Москва, 2024 г.

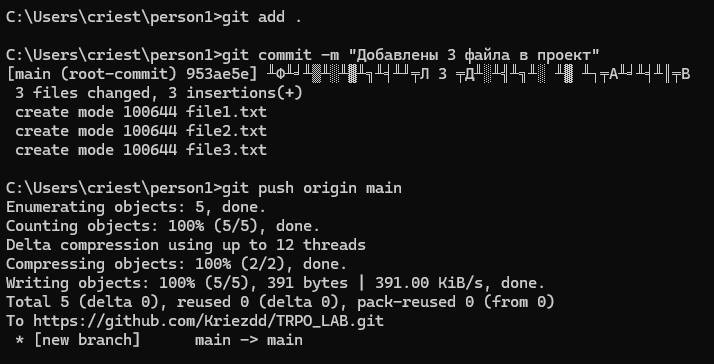
**Практическая работа №1. Система контроля версий Git.**

1. Создать новый Remote Repository (удаленный репозиторий) на GitHub. Указать описание репозитория. Произвести клонирование репозитория на локальную машину в Local Repository и создать LocalWorkingDirectory.

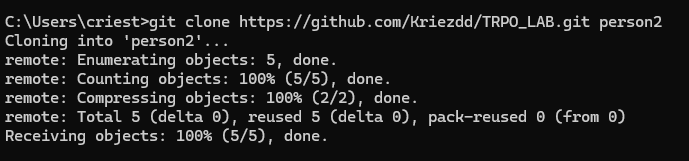


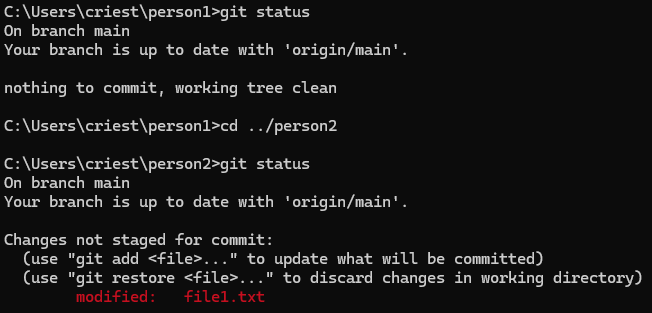
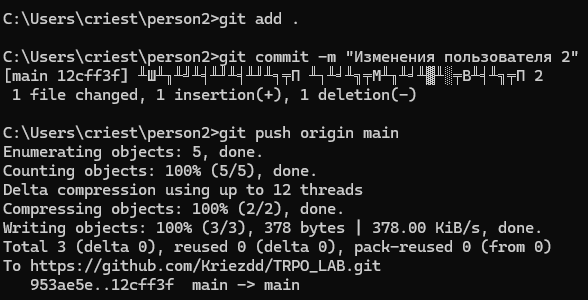


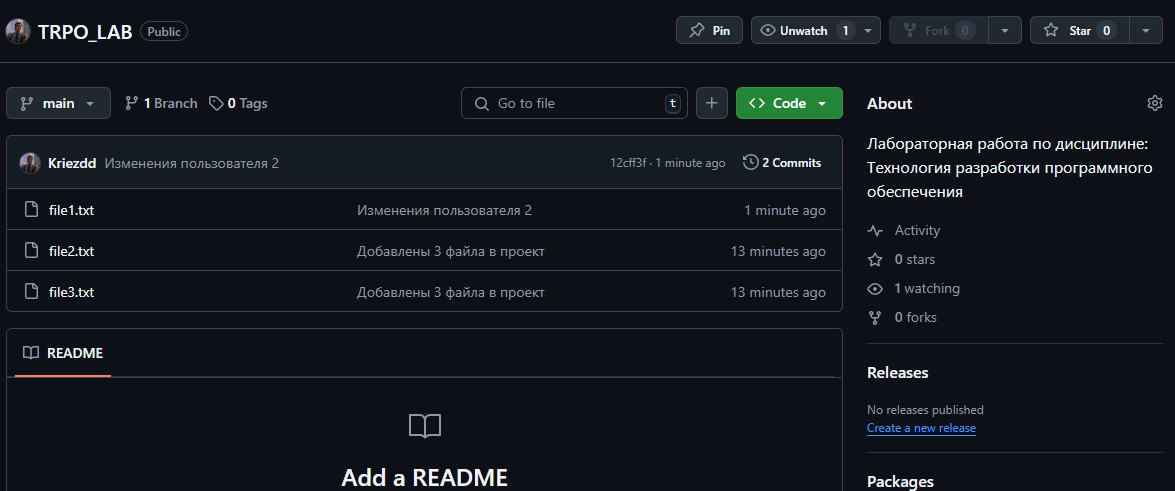
2. В папке LocalWorkingDirectory (назвать «person1») создать приложение с 3-мя файлами на любом языке программирования (возможно работа с текстовыми файлами). Добавить Commit (коммит) в Local Repository (локальный репозиторий), перенести Commit в Remote Repository.  


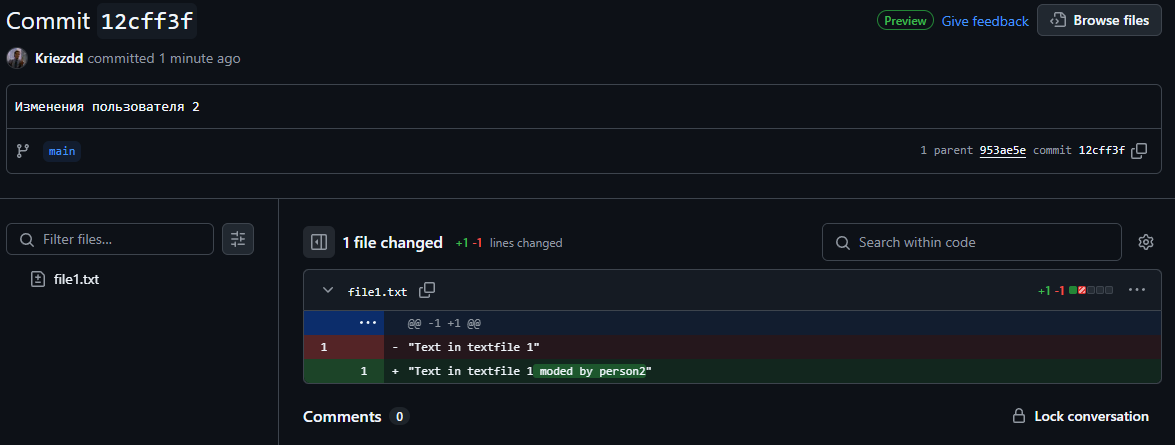


3. Произвести клонирование Remote Repository еще в одну LocalWorkingDirectory (назвать «person2»). В текущей LocalWorkingDirectory добавить в один из файлов новую информацию, добавить Commit в Local Repository. Проверить изменения в LocalWorkingDirectory «person1». Загрузить изменения из LocalWorkingDirectory «person2» в Remote Repository,

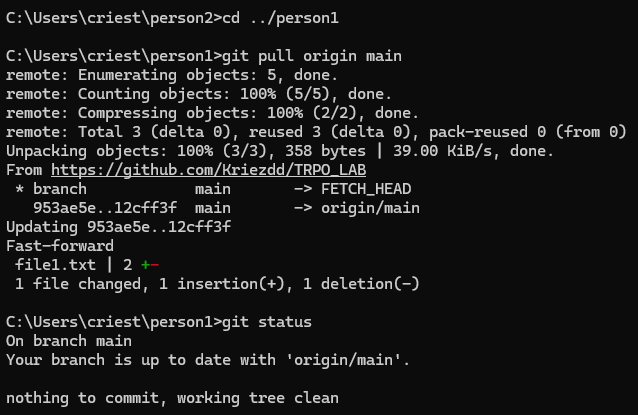


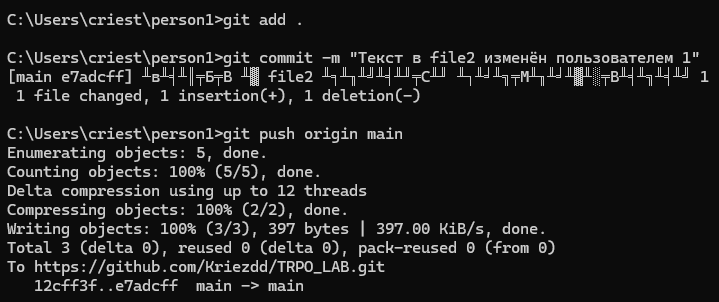
 

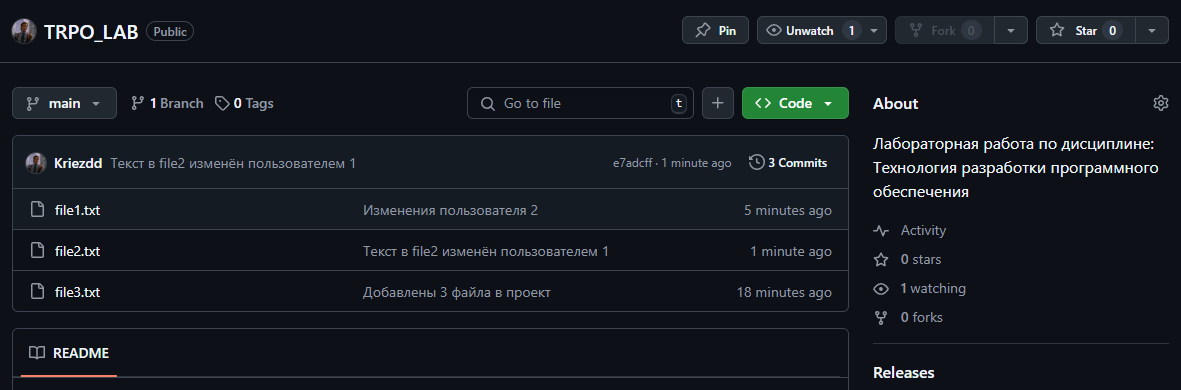


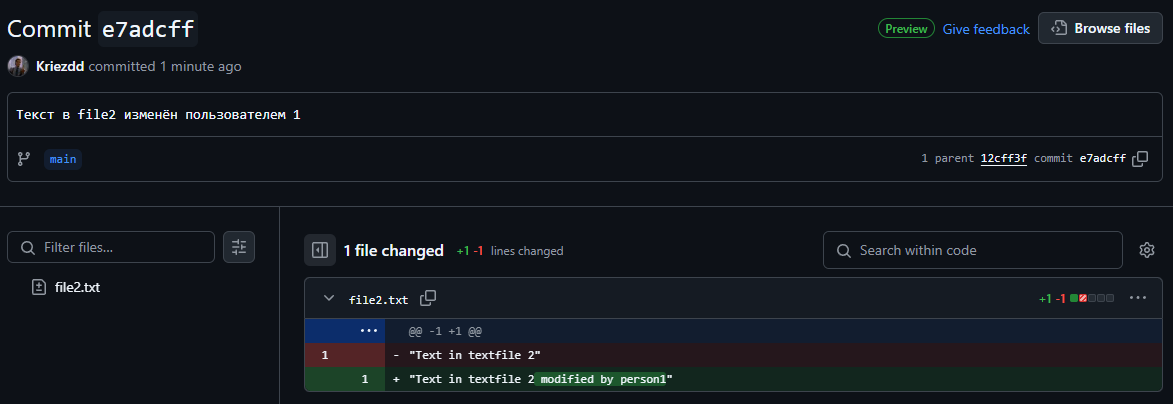


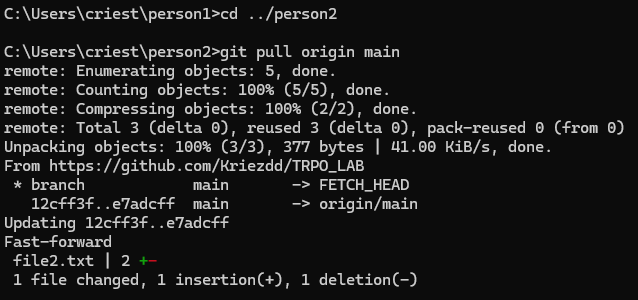
Скачать изменения в LocalWorkingDirectory «person1», проверить изменения в LocalWorkingDirectory «person1». Добавить изменения в другой файл, создать новый Commit, и повторить работу с Remote Repository.



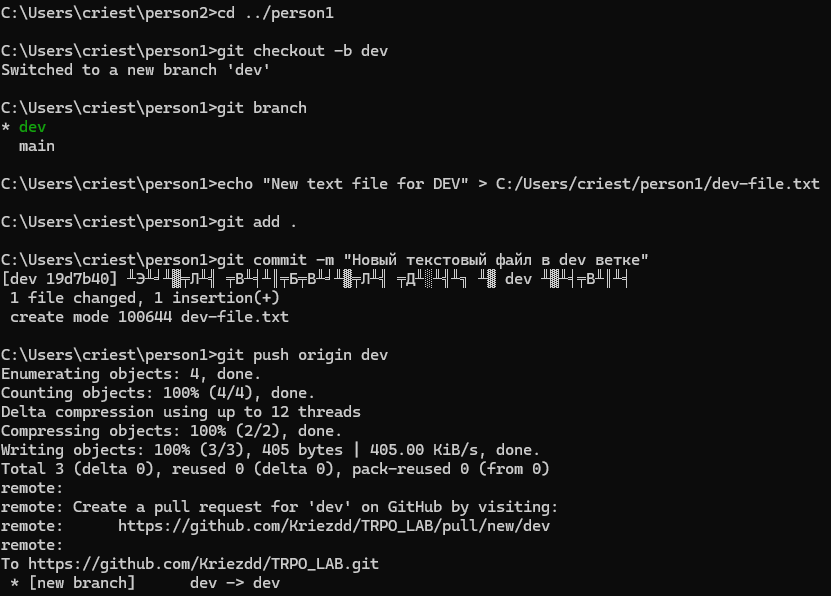


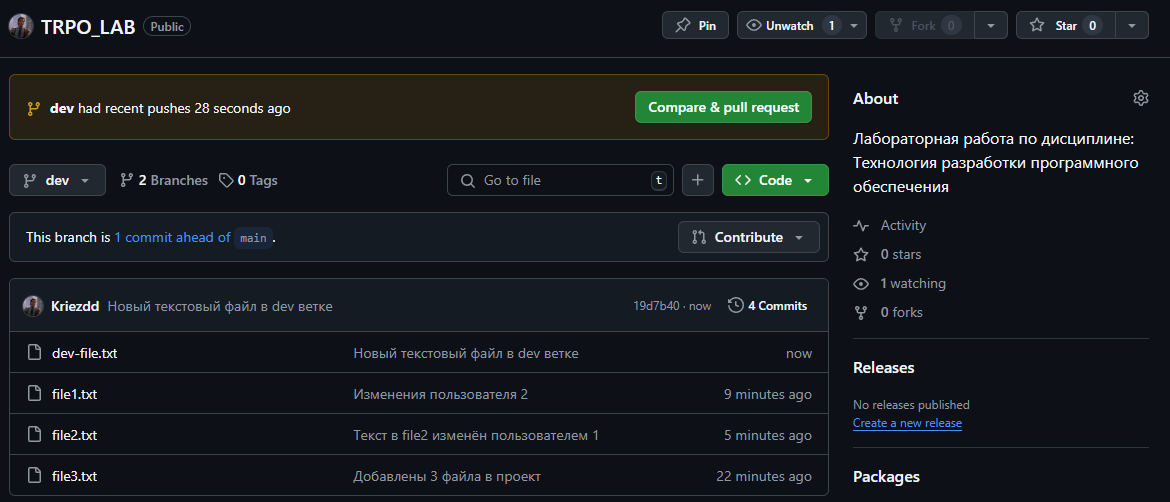


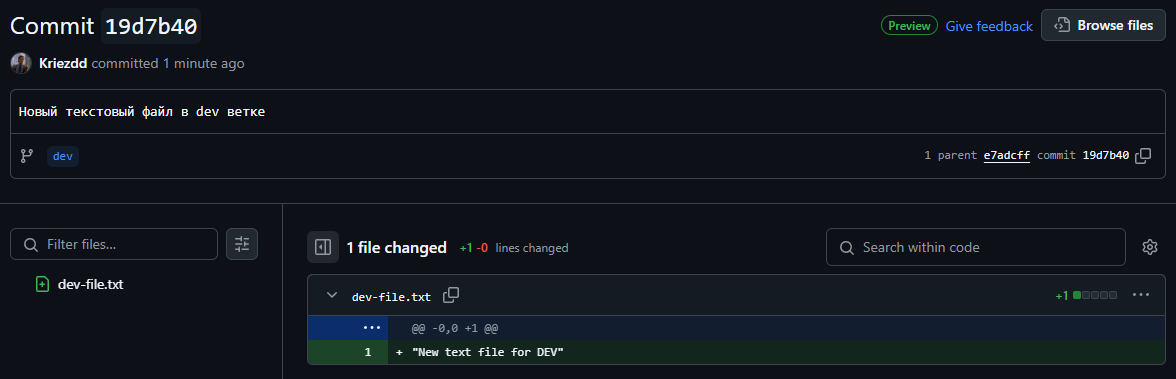




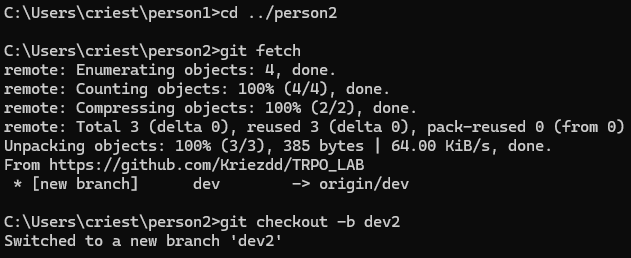
4. Создание новых веток. Для разработки новой функциональности необходимо создать новую ветку в Local Repository «person1» с названием «dev». Переключиться на нее. Создать новый файл в LocalWorkingDirectory «person1» в новой ветке. Сделать новый Commit в ветке и провести обновление в Remote Repository.

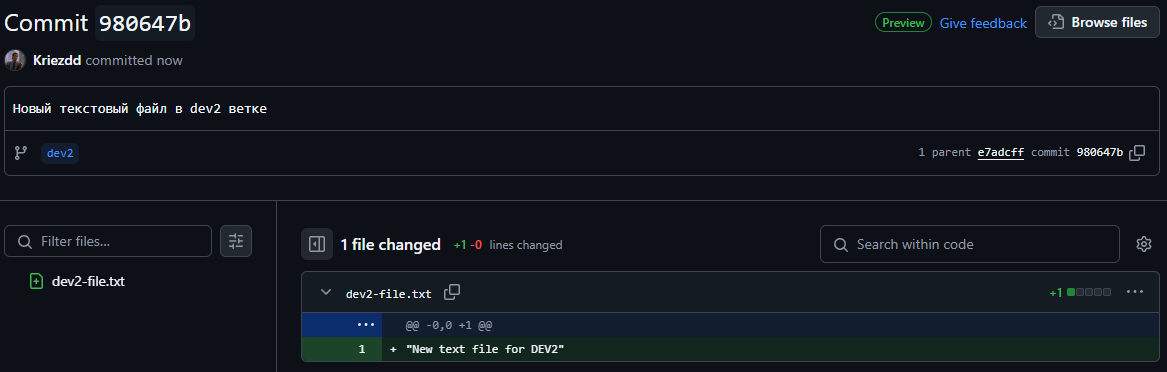




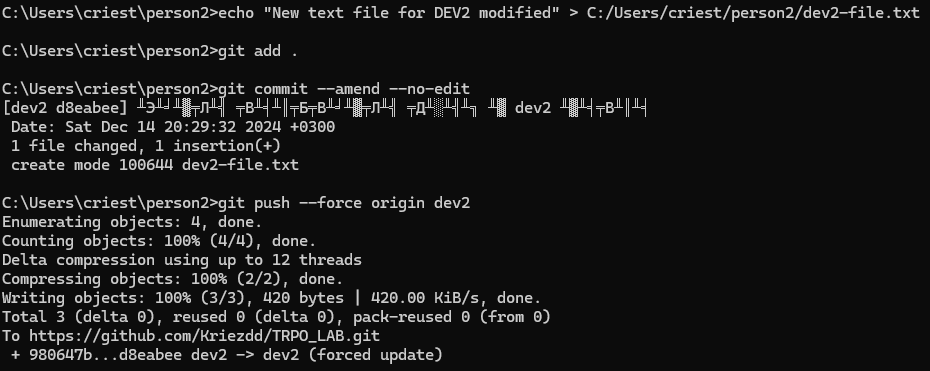


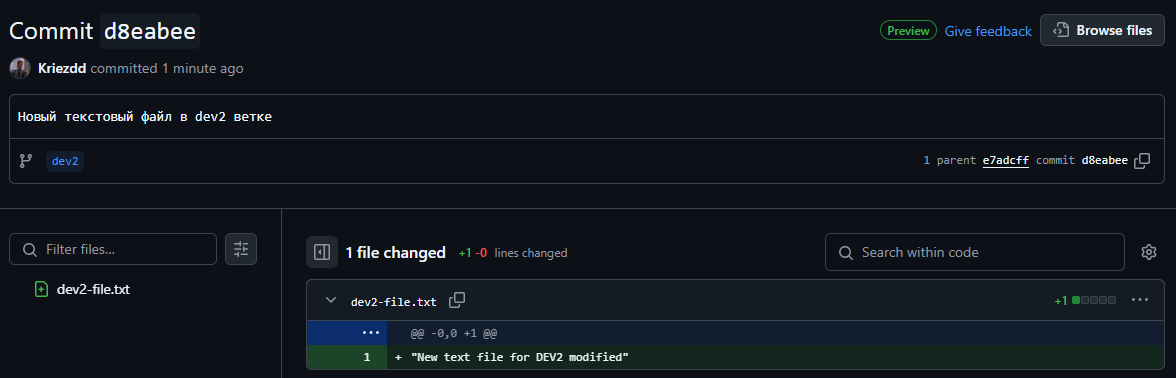
В LocalWorkingDirectory «person2» получить последние обновления из Remote Repository, создать новую ветку «dev2» в которой создать новый файл и выполнить Commit и загрузить его в Remote Repository.

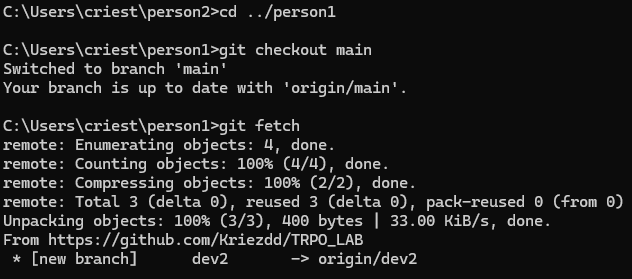


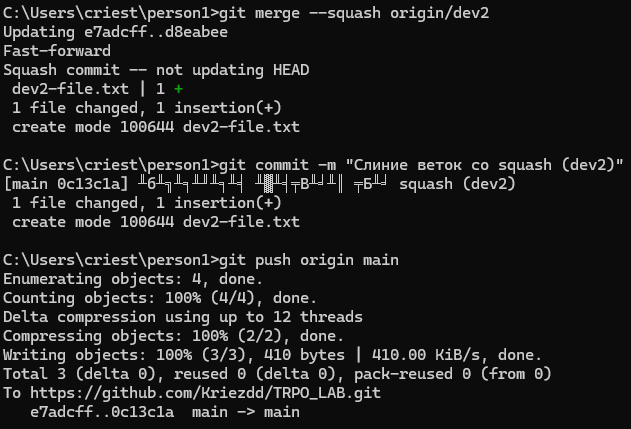
Сделать небольшие изменения в новом файле и добавить эти изменения в предыдущий Commit.

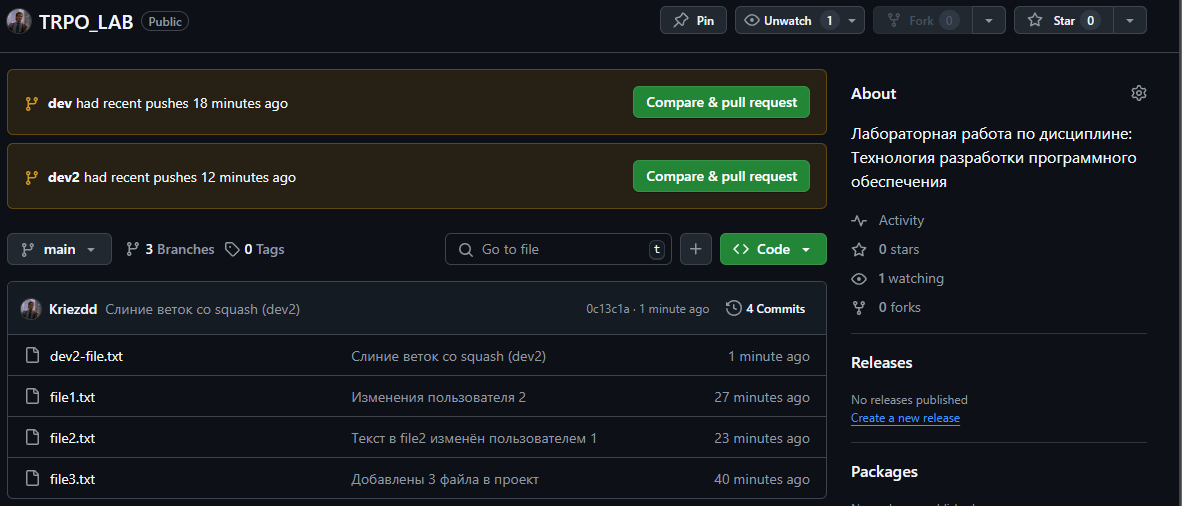


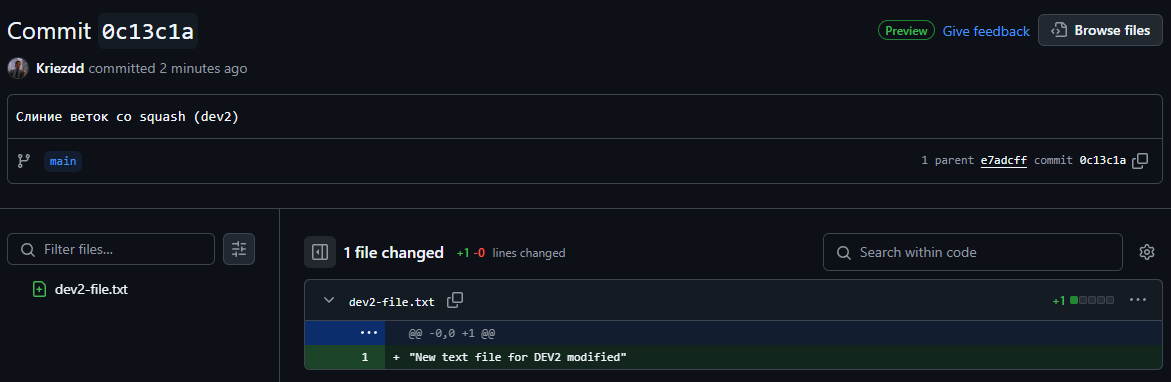
 5. Слияние веток.

5.1 Изучить слияние веток с помощью Squash Commits, Merge, Rebase. Привести примеры слияния и полученные результаты. Слияние веток проводить повторно, выполняя пункт 4.

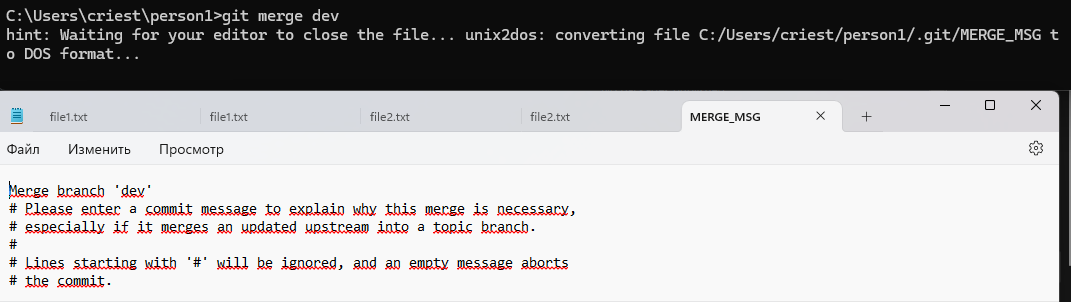
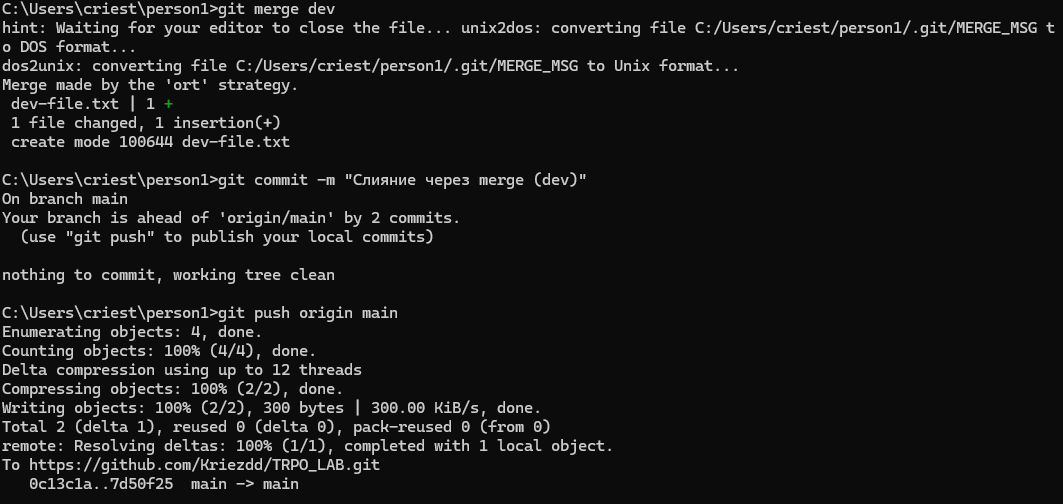
1) Squash Commits

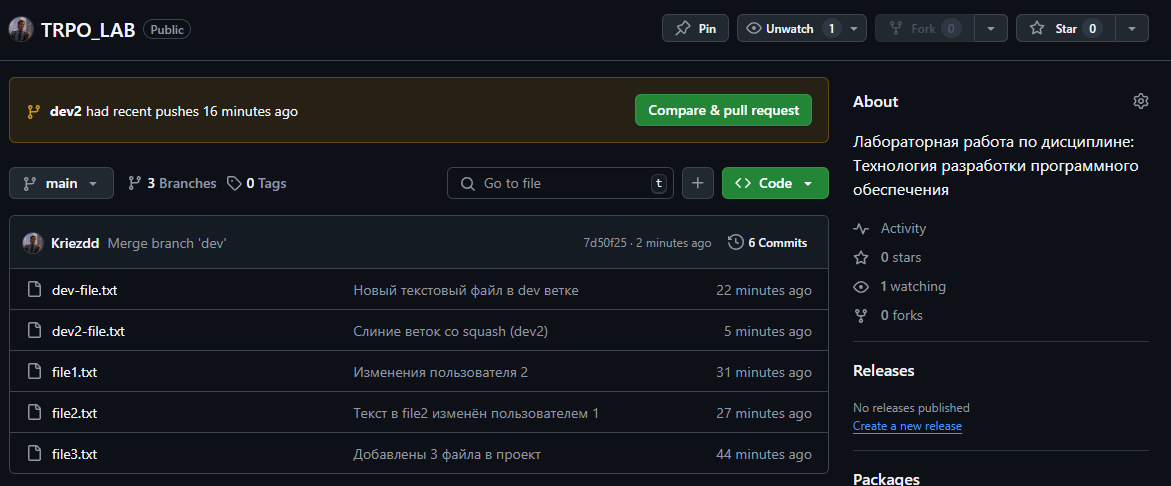
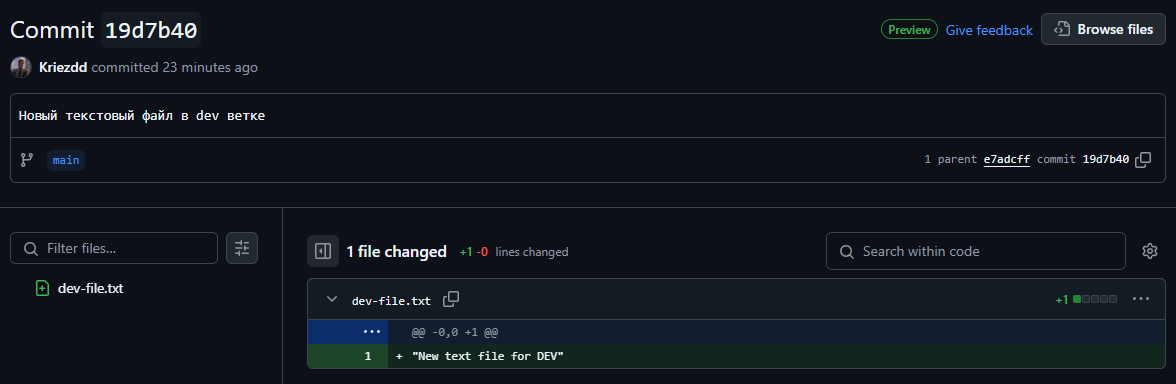




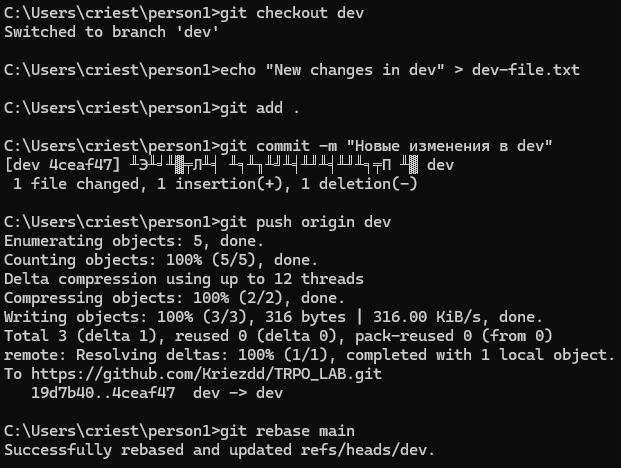
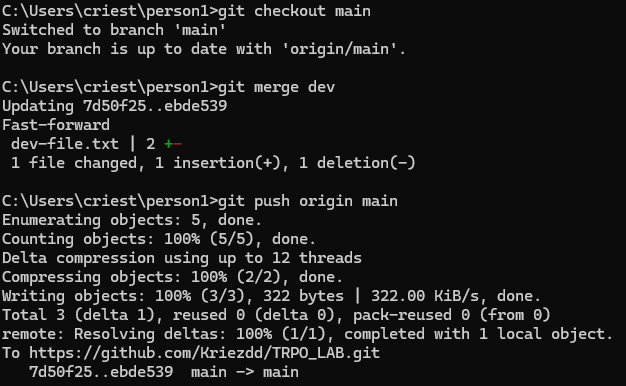


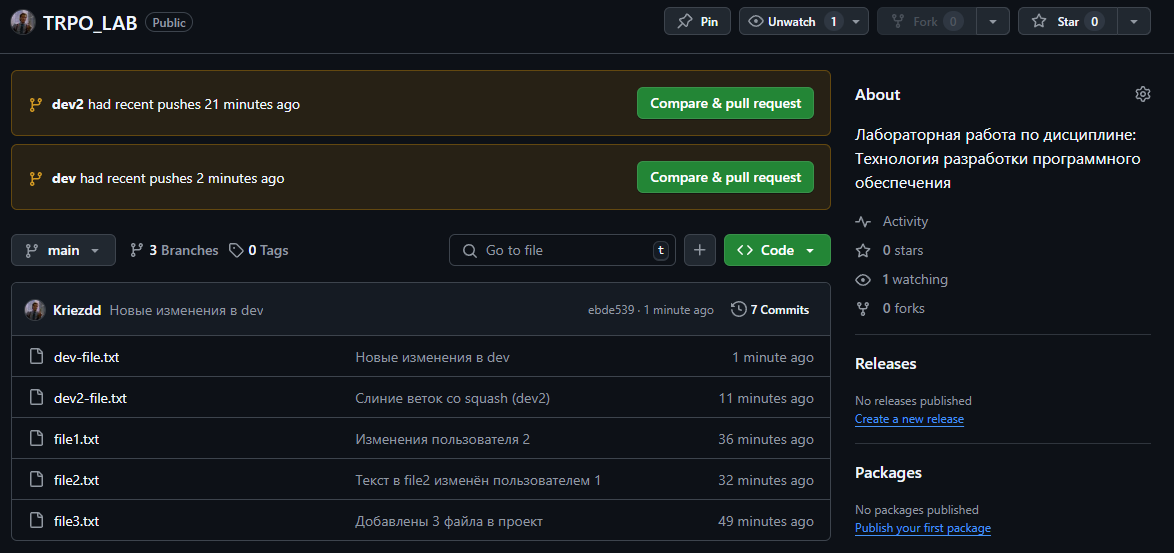
2) Merge

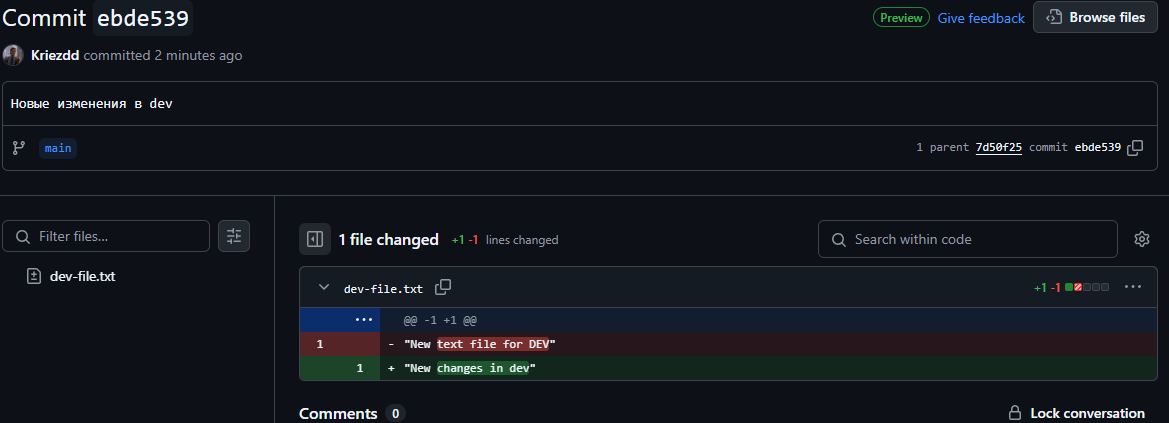
 

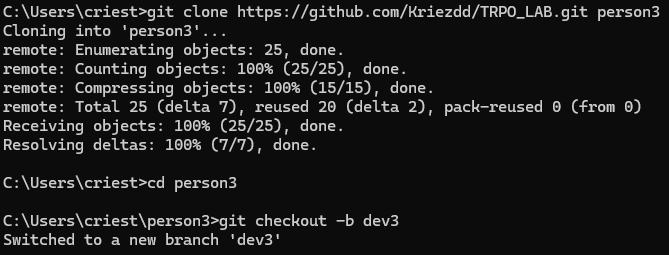
 

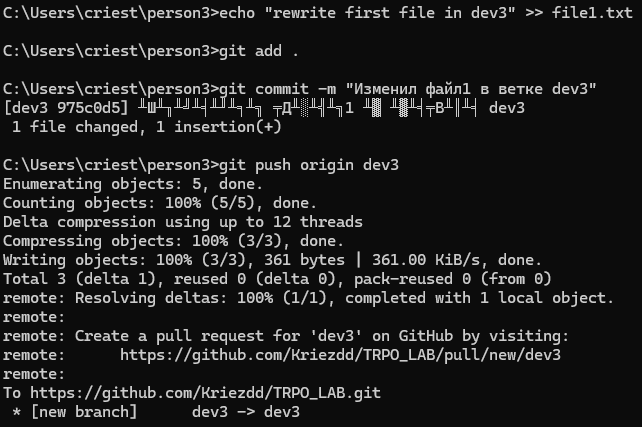
3) Rebase

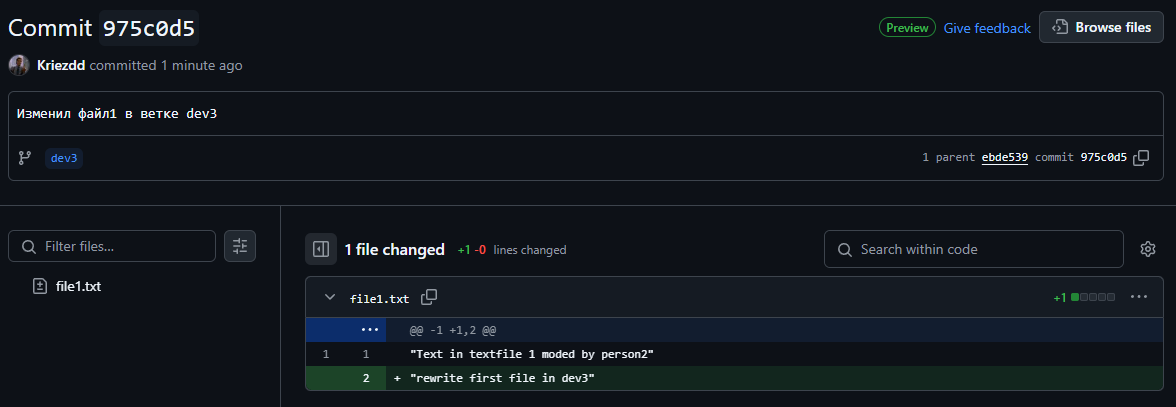
 



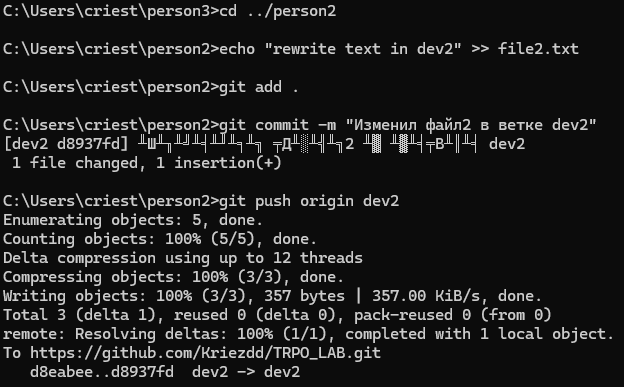


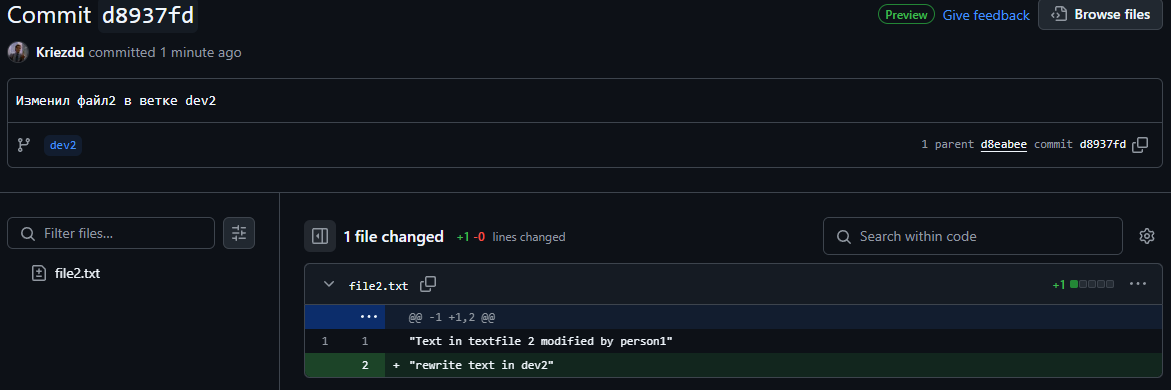
5.2 Разрешение конфликтов слияния. Создать третью LocalWorkingDirectory «person3» и загрузить туда копию из Remote Repository. Для третьей LocalWorkingDirectory создать свою ветку «dev3» и изменить в ней один из первых файлов с последующим созданием Commit и выгрузкой Commit на Remote Repository





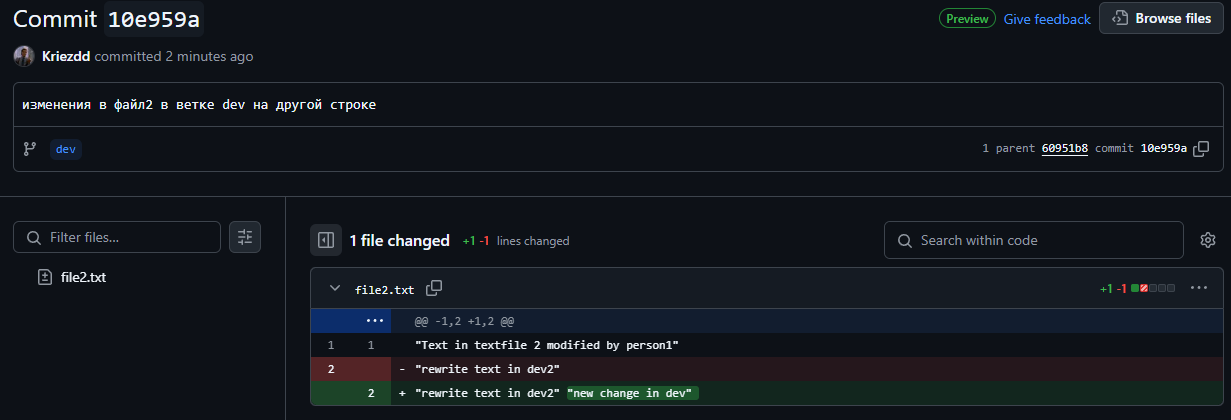
Также необходимо изменить другой файл в LocalWorkingDirectory «person2» в ветке «dev2», создать и выгрузить новый Commit на Remote Repository.



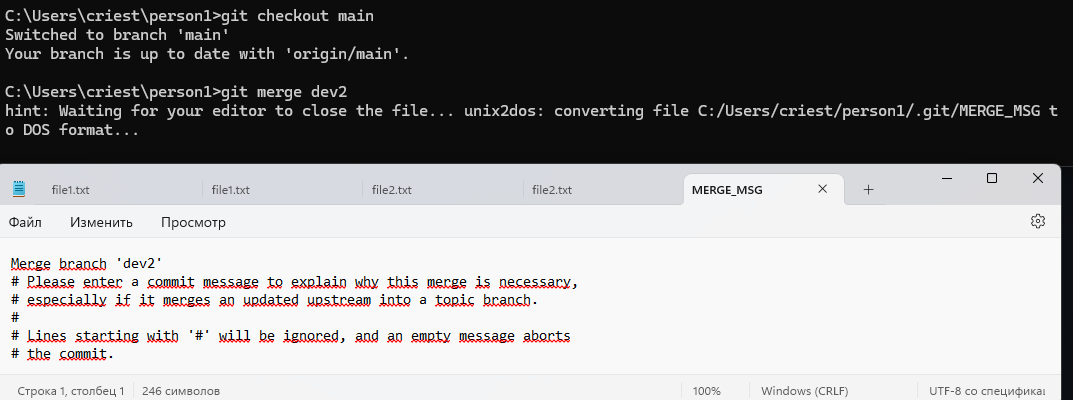
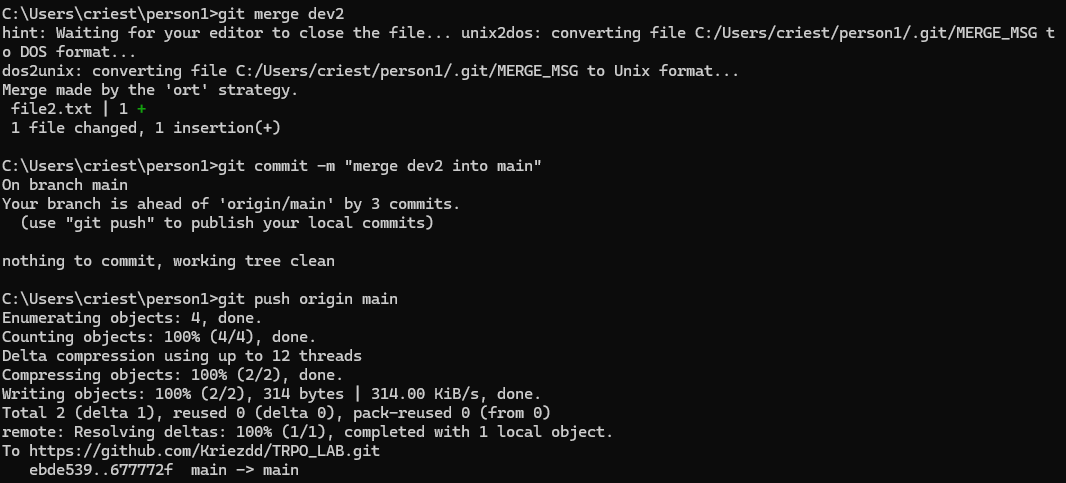
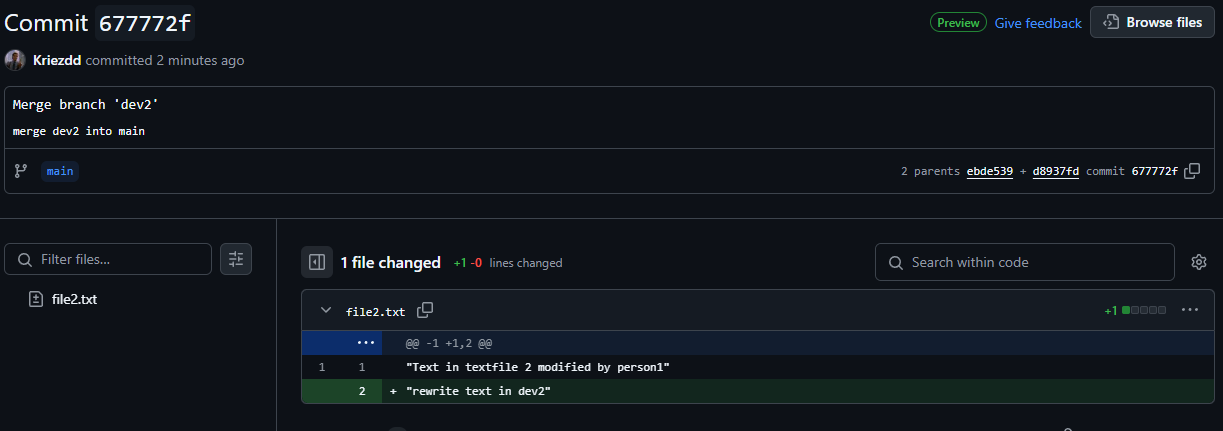


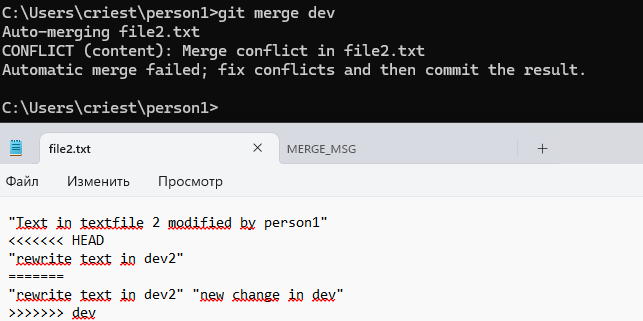
В LocalWorkingDirectory «person1» необходимо изменить тот же файл, что и в третьей LocalWorkingDirectory, но в другом месте и добавить Commit в ветку «dev1».

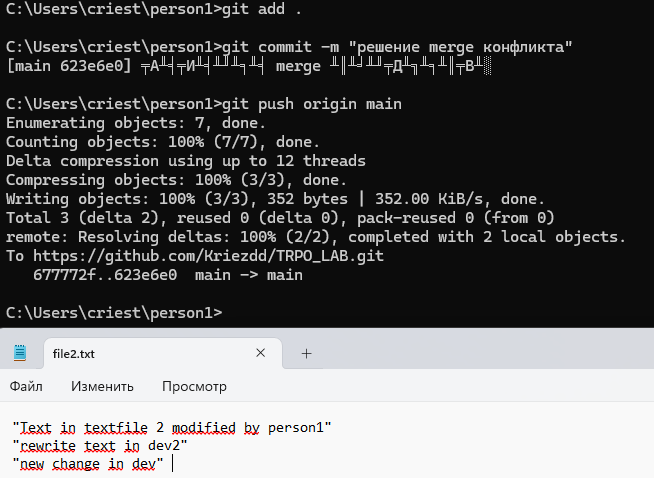


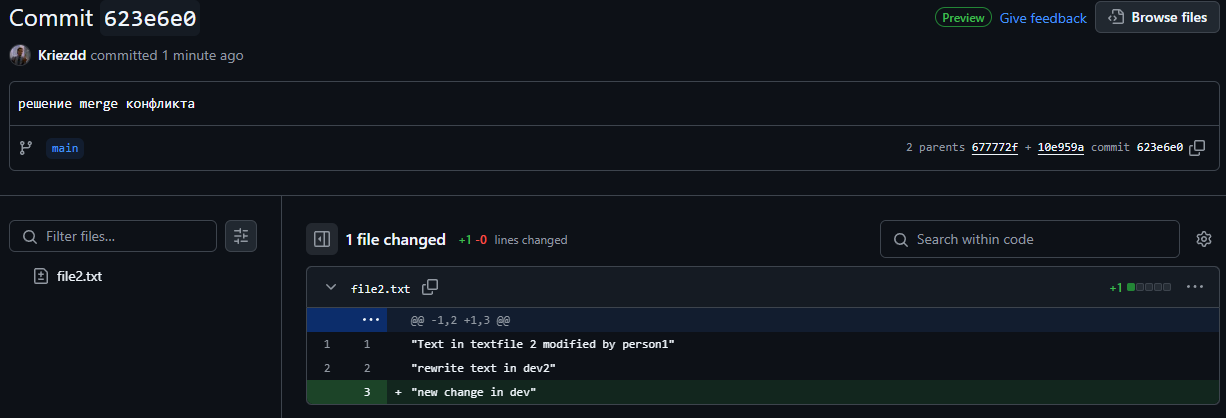


Данные действия имитируют работу трех программистов над разными задачами и одинаковыми или различными файлами проекта. Поочередно проведите слияние веток «dev» с веткой «master» и опишите результаты. Рассмотрите методы исправления конфликтов слияния.

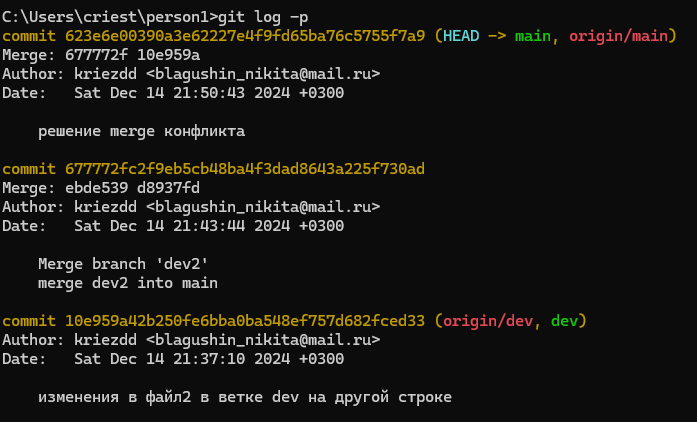
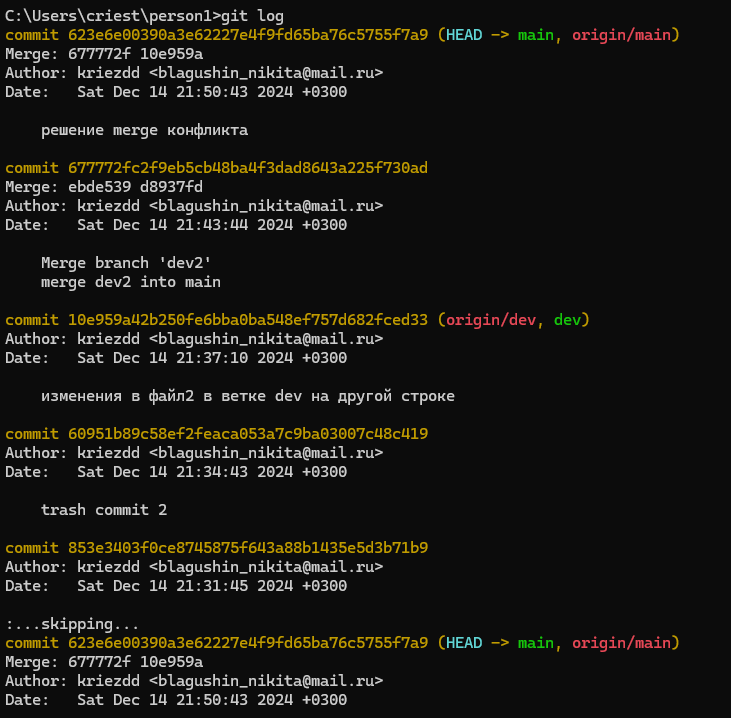
  

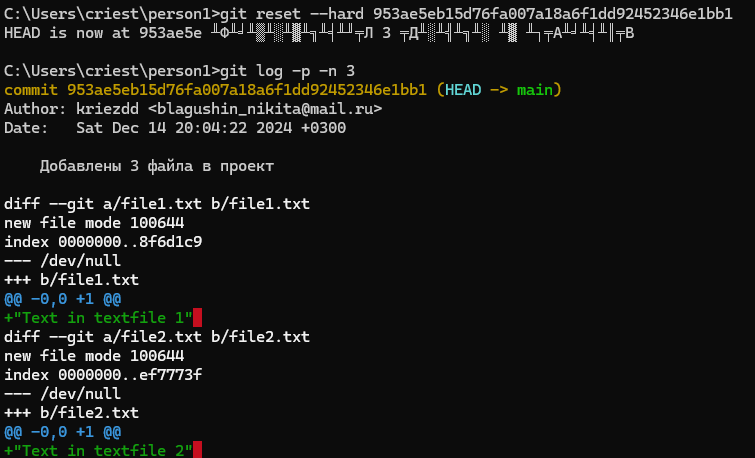




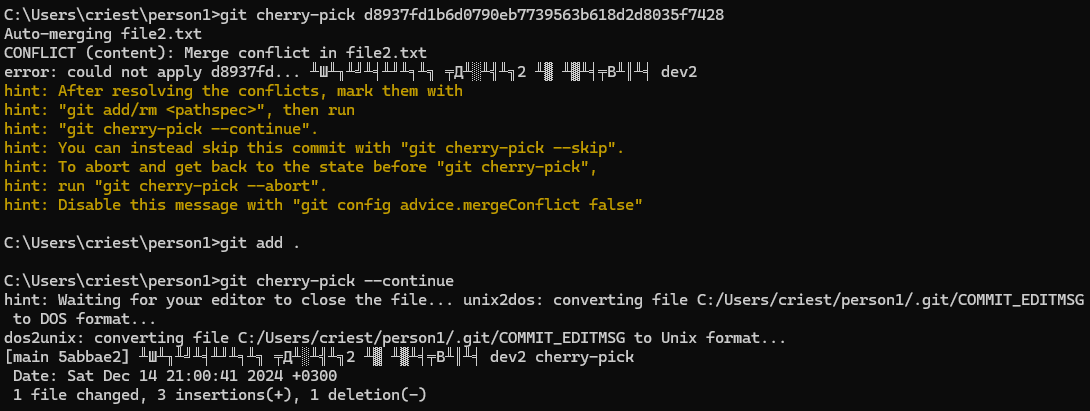


6. Использовать git log для нахождения хэша Commit. Откатить изменения до выбранного Commit а с помощью git reset --hard <commit\_hash>. Для просмотра изменений в каждом Commit можно использовать git log –p





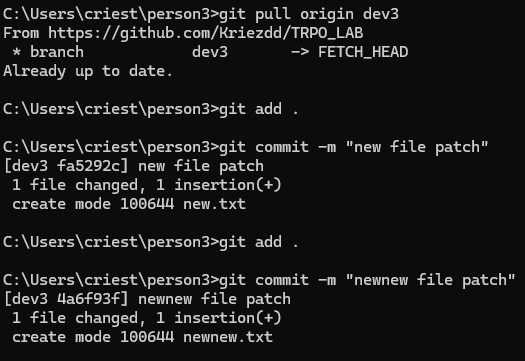
7.Используйте команду git cherry-pick <commit\_hash> для перемещения Commit из ветки «dev2» на текущую ветку.

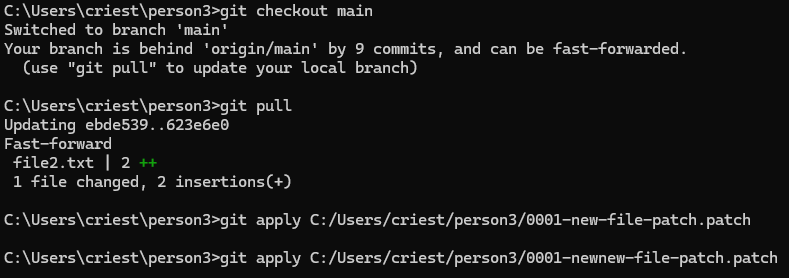


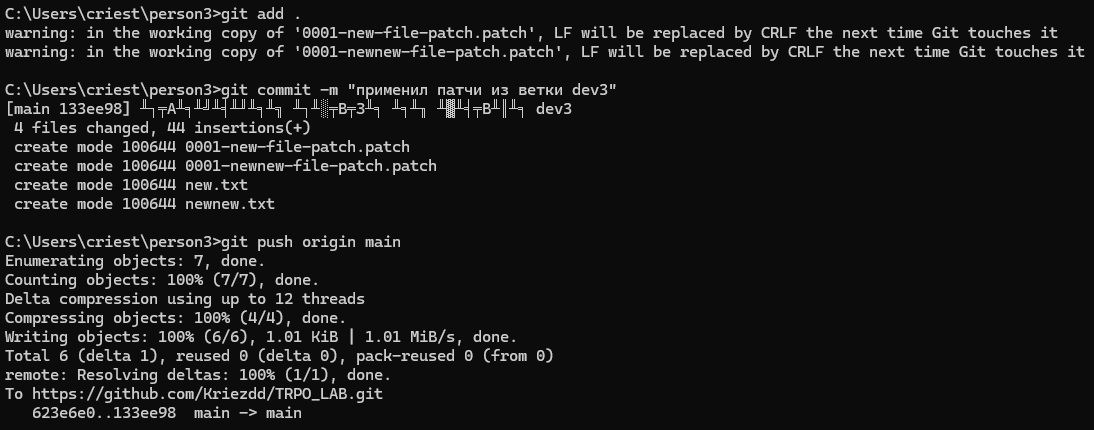
Используйте git rebase для перебазирования Commit ветки «dev1» на другую ветку. Для перемещения нескольких Commit можно использовать git rebase -i <commit\_hash>.



Создайте серию коммитов в ветке «dev3», создайте патчи для этих коммитов с помощью git format-patch и примените их к ветке main с помощью git apply

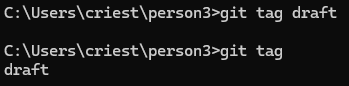
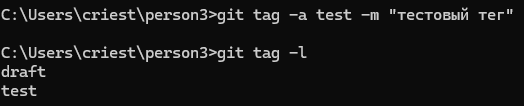
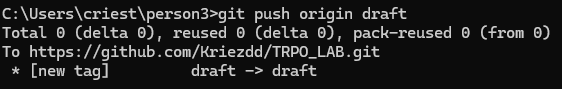
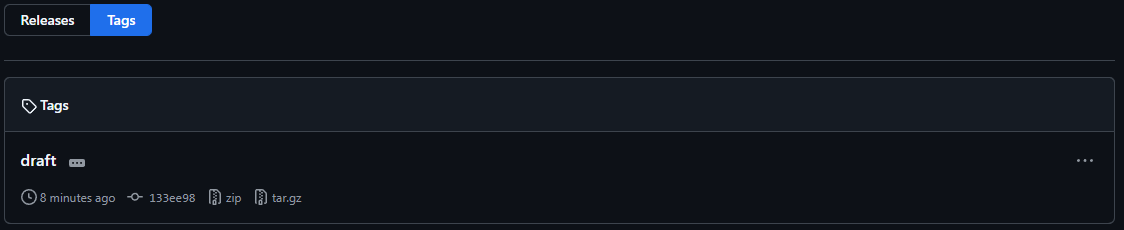
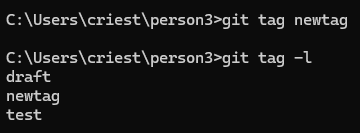






**Вариант 1.**

«Создание тегов для версионирования кода»:

* 1. Используя команду git tag <tag\_name>, создайте легковесный тег для текущего состояния репозитория.
  2. Проверьте созданный тег с помощью команды git tag.  
       
     
  3. Для создания аннотированного тега с сообщением используйте команду git tag -a <tag\_name> -m "Message".
  4. Просмотрите список всех тегов с помощью команды git tag -l.  
       
     
  5. Публикуйте созданный тег на удаленном репозитории с помощью команды git push origin <tag\_name>.  
       
     
  6. Убедитесь, что тег успешно опубликован, проверив его на удаленном репозитории.  
       
     
  7. Создайте еще один легковесный тег с другим именем с использованием git tag.
  8. Просмотрите изменения в тегах после создания нового тега с помощью git tag -l.  
       
     
  9. Отправьте оба тега на удаленный репозиторий с помощью команды git push origin –tags.
  10. Проверьте наличие всех тегов на удаленном репозитории с помощью git ls-remote –tags origin.  
      