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 №4

По дисциплине: «Системы хранения и обработки данных»

Тема: «Основы работы с системой управления версиями Git»

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

подпись, дата

Принял: Короленко В.В.

подпись, дата

Воронеж 2024

Лабораторная работа №4

«Основы работы с системой управления версиями Git»

1. Цель лабораторной работы: изучить назначение распределенной системы управления версиями, освоить процесс создания репозитория и основы управления версиями файлов.

2. Задачи лабораторной работы:

- создание профиля и репозитория на платформе github;

- установка git, запуск;

- создание ветки для отработки своего проекта;

- изучение основных команд git;

- клонирование удаленного репозитория на локальный компьютер;

- создание файла с данными о себе, изменение файла и отправка соответствующих изменений из локального репозитория на удаленный;

- откат изменений до предыдущего коммита.

3. Результаты лабораторной работы

Зайдем на сайт github.com, зарегистрируемся и авторизируемся. Создадим новый репозиторий. Для этого необходимо:

- нажать на кнопку «+» в правом верхнем углу и выбрать «New repository»;

- ввести название репозитория;

- выбрать публичный доступ «Public»;

- поставить галочку для создания файла README.md;

- нажать зеленую кнопку «Create repository».

Результат выполнения данных действий представлен на рисунках 1-2.

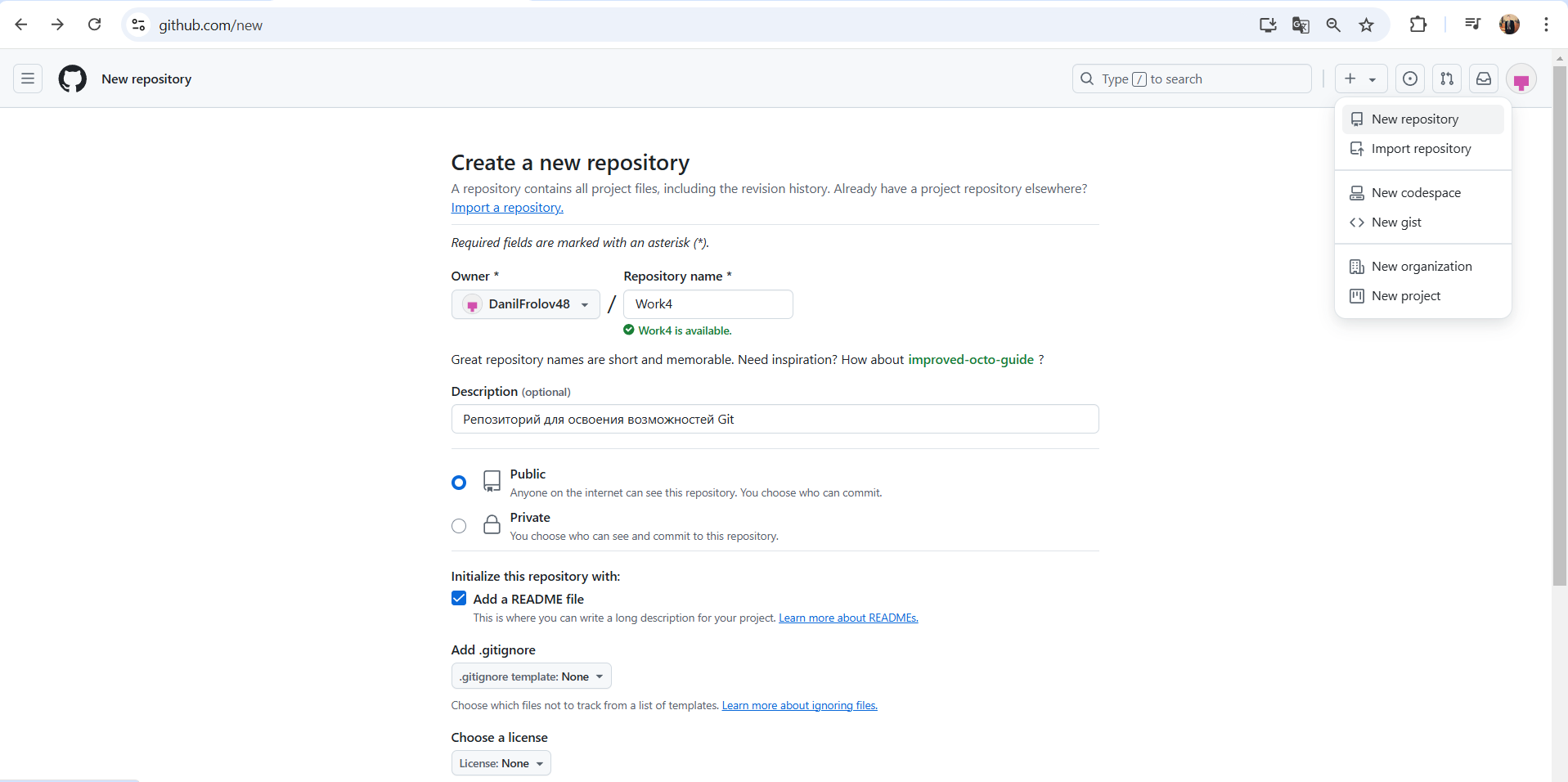


Рисунок 1 – Создание репозитория

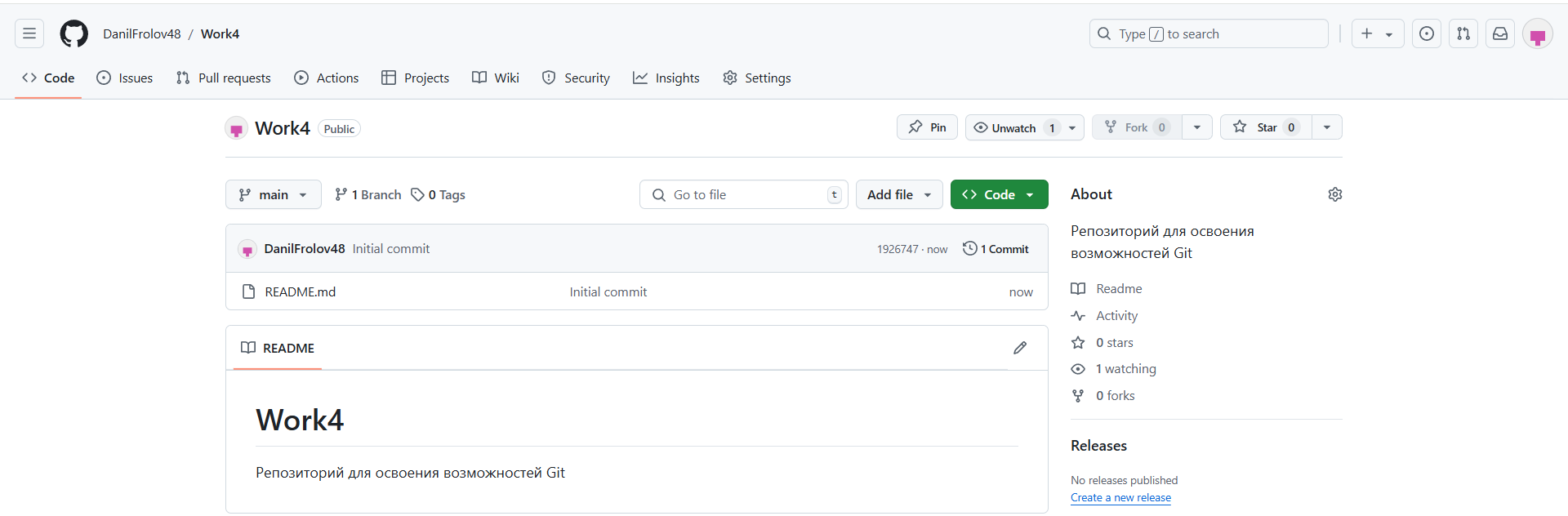


Рисунок 2 – Созданный репозиторий

Добавим комментарий в файл readme.md о создании репозитория, укажем цель лабораторной работы, фамилию, имя и отчество студента. Для этого необходимо нажать на значок карандаша «Edit file», внести соответствующую информацию в файл и нажать на зеленую кнопку «Commit changes». Результат представлен на рисунках 3-5

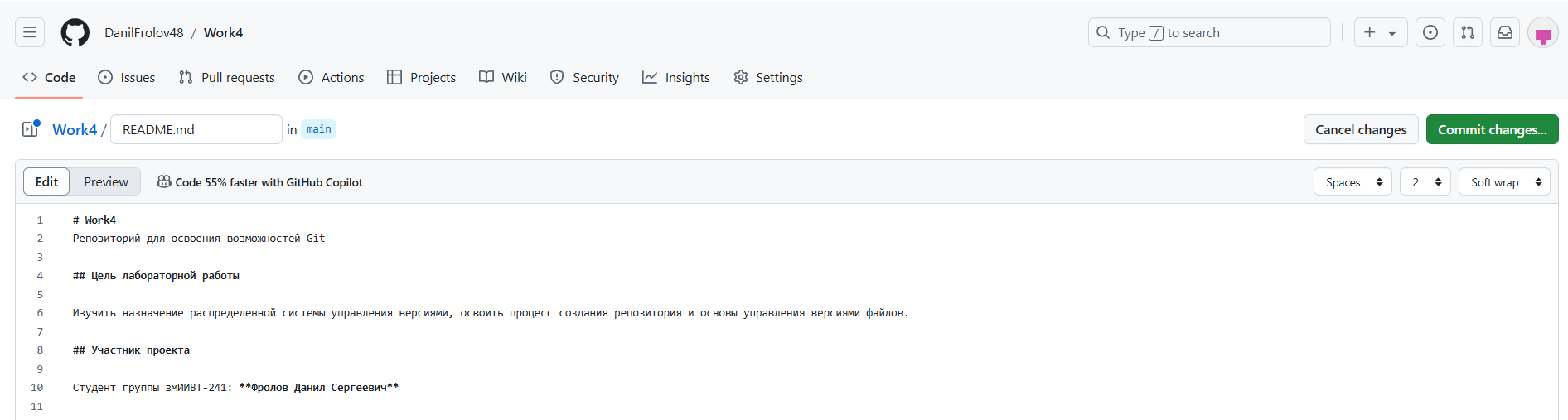


Рисунок 3 – Добавление комментария в файл «readme.md»

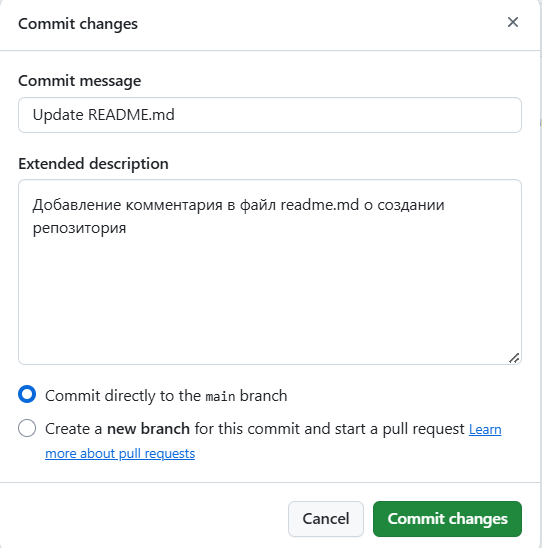


Рисунок 4 – Окно «Commit changes»

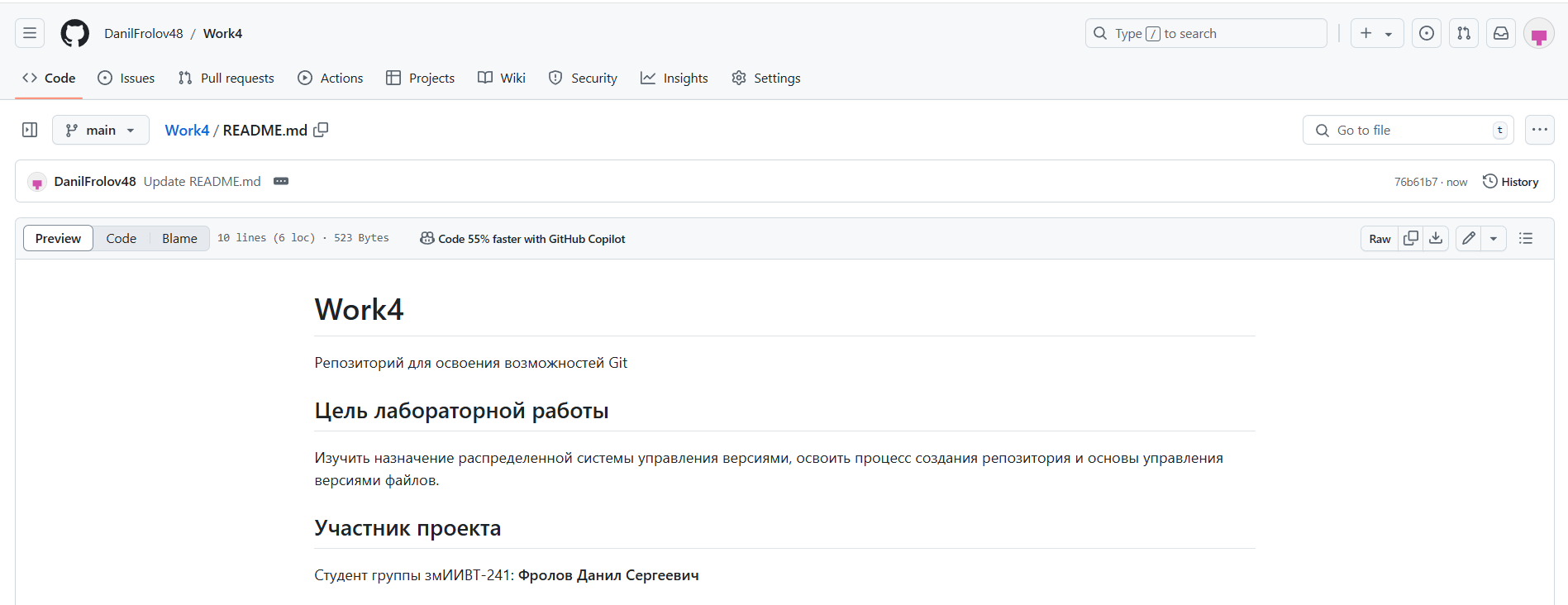


Рисунок 5 – Результат создания комментария в «readme.md»

Скачаем и установим git на локальный компьютер. После этого запустим его (рисунок 6).



Рисунок 6 – Запуск git

Создадим новую ветку «LR4-FDS» для выполнения лабораторной работы, которая будет ответвлением от главной ветки. Для этого на GitHub перейдем на главную страницу репозитория. В представлении дерева файлов слева выберем раскрывающееся меню ветвей, затем нажмем «View all branches» (рисунок 7).

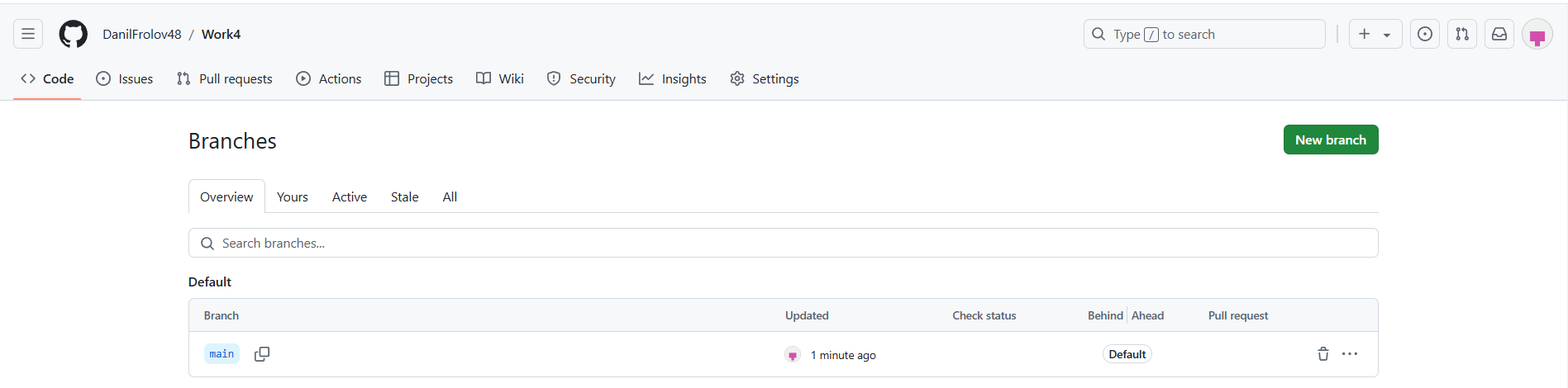


Рисунок 7 – Просмотр всех ветвей

Нажмем на зеленую кнопку «New branch» и введем имя ветви. В разделе «Source» выберем источник для нашей ветви – ветку «main» и нажмем зеленую кнопку «Create new branch». Результат представлен на рисунках 8-9.

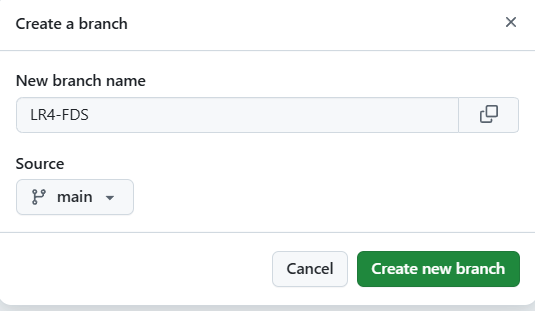


Рисунок 8 – Создание ветки «LR4-FDS»

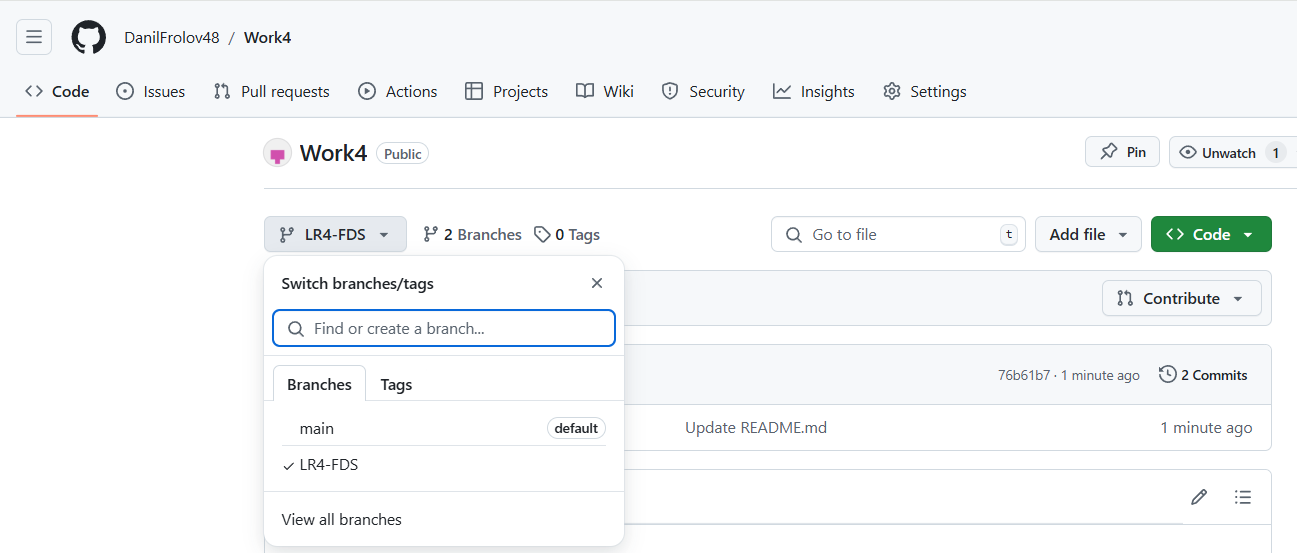


Рисунок 9 – Результат создания новой ветки и переключение на нее

Изучим основные команды git, опишем их в файле «readme.md» в своей ветке. Результат представлен на рисунке 10.

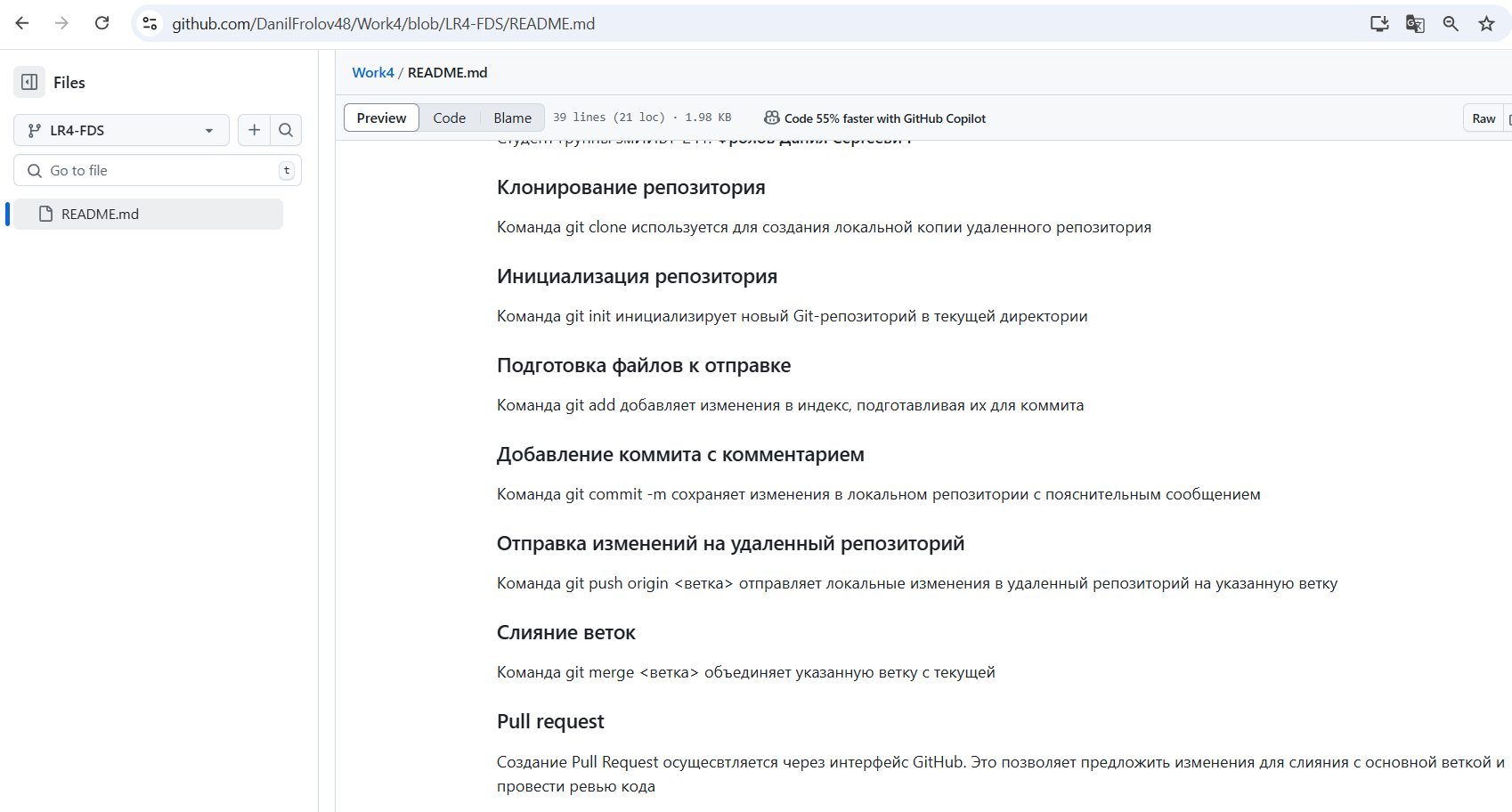


Рисунок 10 – Описание основных команд git в файле «readme.md» в своей ветке

Клонируем удаленный репозиторий на локальный компьютер. Для этого необходимо в GitHub скопировать ссылку на репозиторий (рисунок 11).

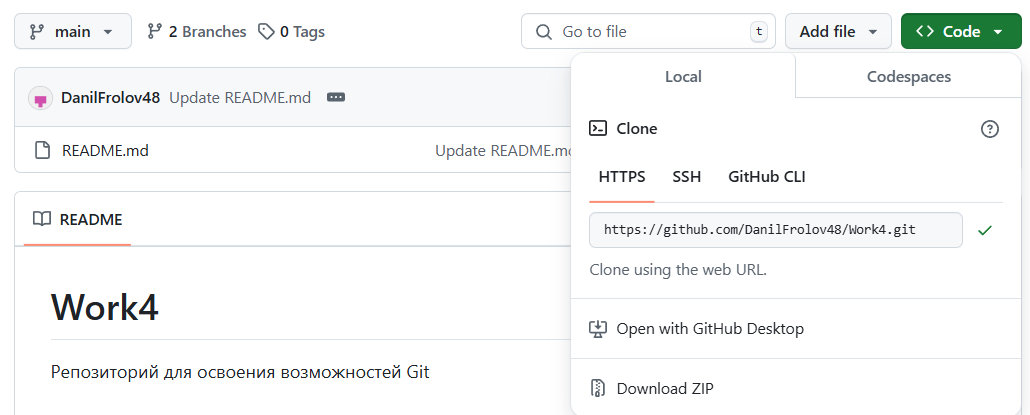


Рисунок 11 – Копирование ссылки на репозиторий

Далее создадим папку на локальном компьютере «LR\_GIT». Открываем терминал Git Bash и выполняем команду cd C:\Users\User\LR\_GIT. Затем выполняем команду git clone <https://github.com/DanilFrolov48/Work4.git>. После выполнения команды git clone в папке LR\_GIT появится новая подпапка с именем репозитория. Переходим в папку Work4 и теперь находимся в локальном клоне нашего репозитория. Результат выполнения представлен на рисунке 12.

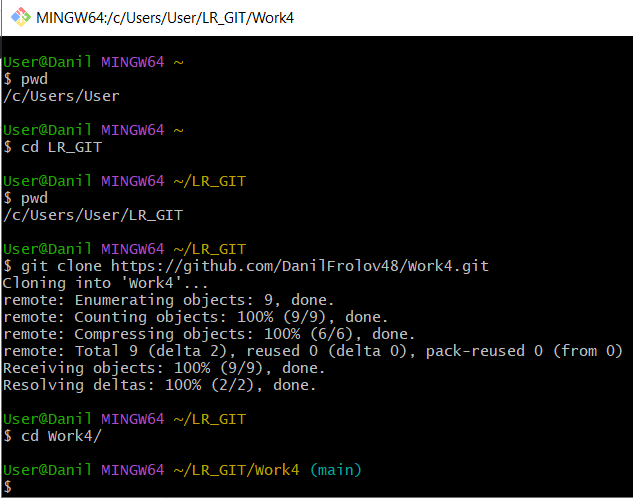


Рисунок 12 – Клонирование удаленного репозитория на локальный компьютер

Создадим в локальном репозитории текстовый файл «FDS.txt» с данными о себе (рисунок 13).

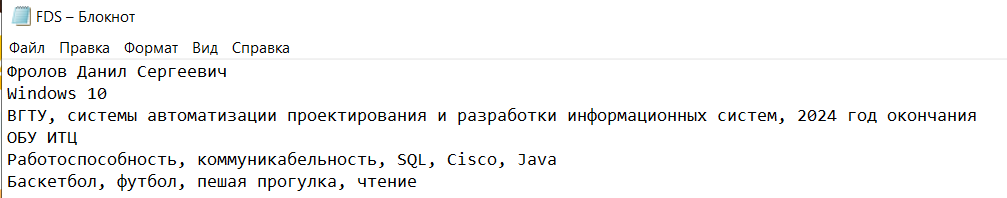


Рисунок 13 – Создание в локальном репозитории текстового файла

Далее необходимо сохранить изменения (работаем в ветке «LR4-FDS») и отправить изменения из локального репозитория на удаленный с использованием ранее изученных команд.

Для этого сначала надо добавить в исключения рабочую директорию репозитория (в качестве защищенной директории). Далее создадим новую ветку «LR4-FDS» и перейдем в нее. Далее подготовим изменения к отправке и сделаем коммит с сообщением «Созданный локально репозиторий связан с удаленным, соответствующие изменения переданы из локального в удаленный репозиторий». Далее синхронизируем локальную ветку с удаленной, используя команды «git fetch origin» и «git merge origin/LR4-FDS». Далее настроим связь с удаленным репозиторием с помощью команды «git remote add origin [https://github.com/DanilFrolov48/Work4.git»](https://github.com/DanilFrolov48/Work4.git). Далее назначим в качестве ветки для синхронизации нужную нам ветку «LR4-FDS» с помощью команды «$ git push --set-upstream origin LR4-FDS». Результат выполнения данных действий представлен на рисунке 14.

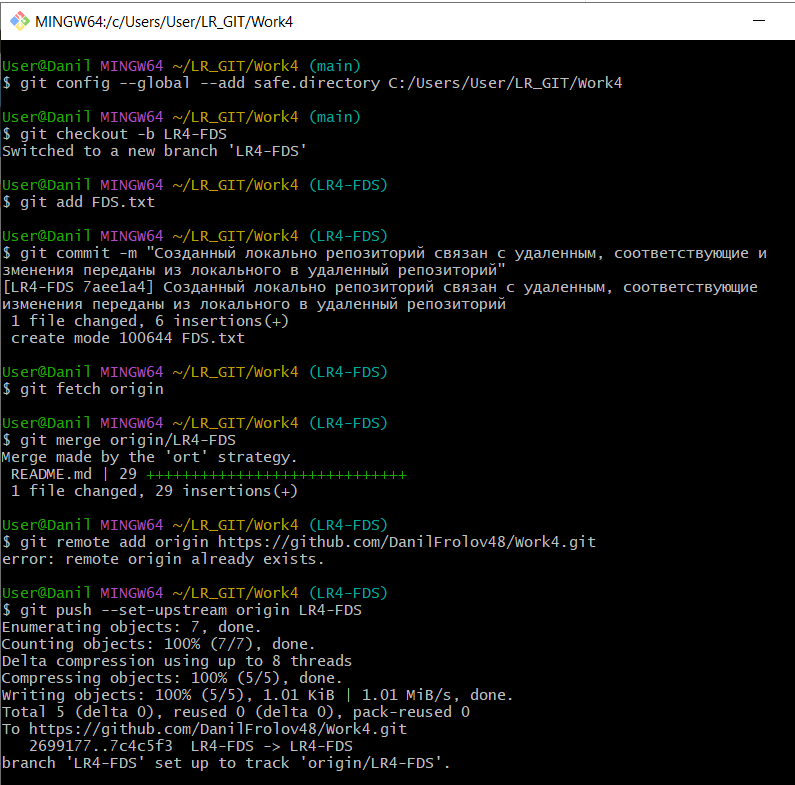


Рисунок 14 – Отправление изменений из локального репозитория в удаленный

Обновим окно с сайтом github в браузере и увидим, что изменения появились (рисунок 15).

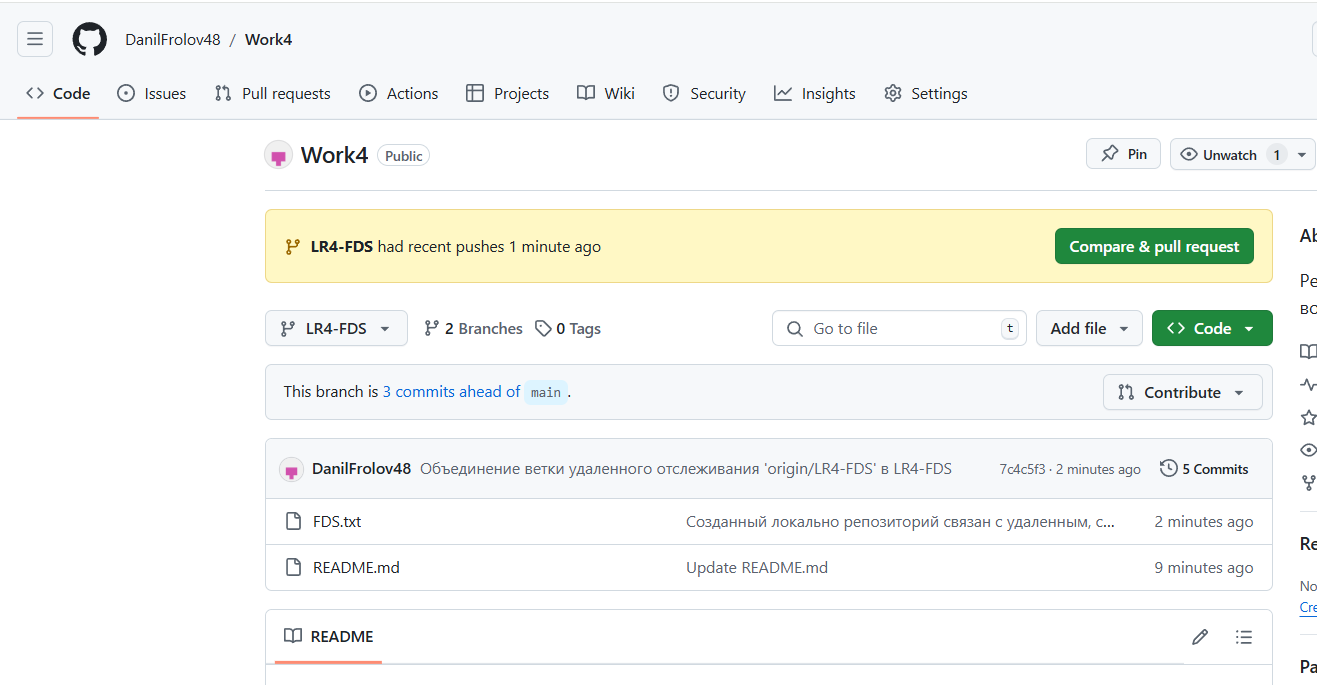


Рисунок 15 – Появление изменений

Добавим в ранее созданный файл «FDS.txt» данные о среднем балле по предыдущему образованию. Отправим изменения на удаленный репозиторий. Результат представлен на рисунках 16-17.

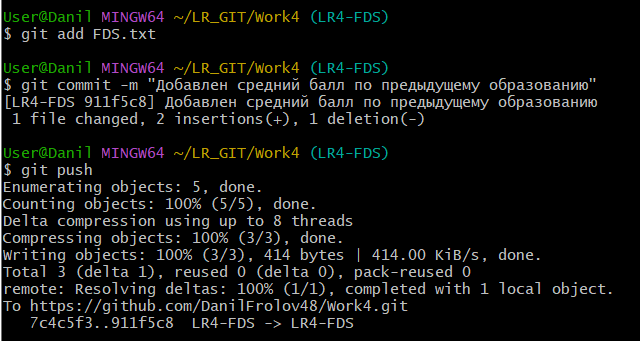


Рисунок 16 – Отправление изменений на удаленный репозиторий

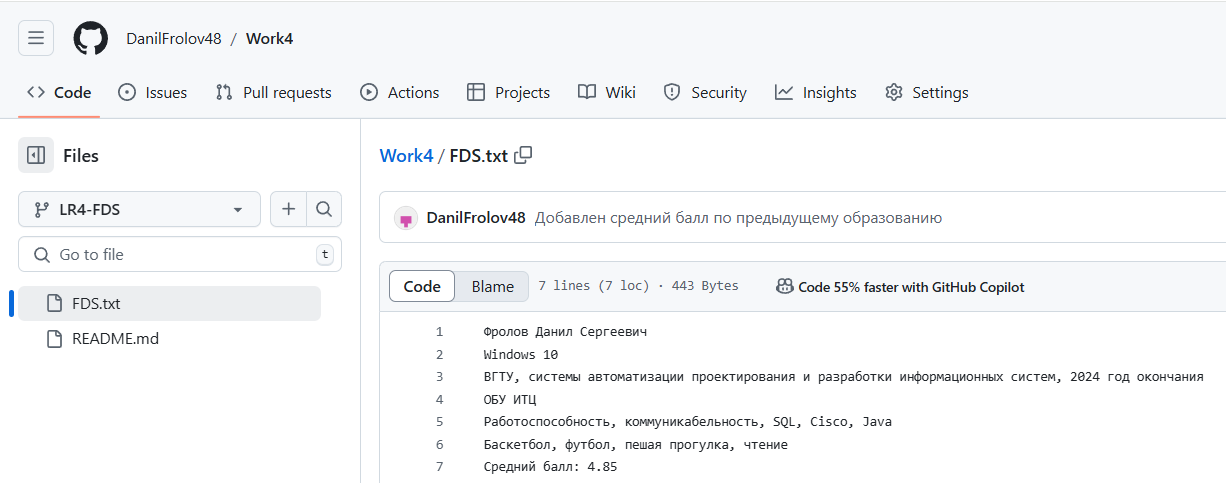


Рисунок 17 – Отображение изменений в файле «FDS.txt»

Добавим в ранее созданный файл «FDS.txt» данные о месте рождения. Отправим изменения на удаленный репозиторий. Результат представлен на рисунках 18-19.

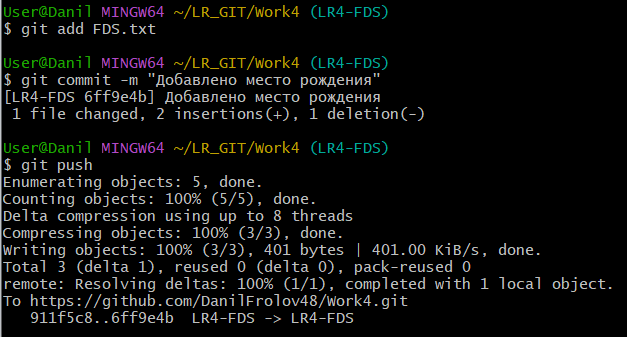


Рисунок 18 – Отправление изменений на удаленный репозиторий

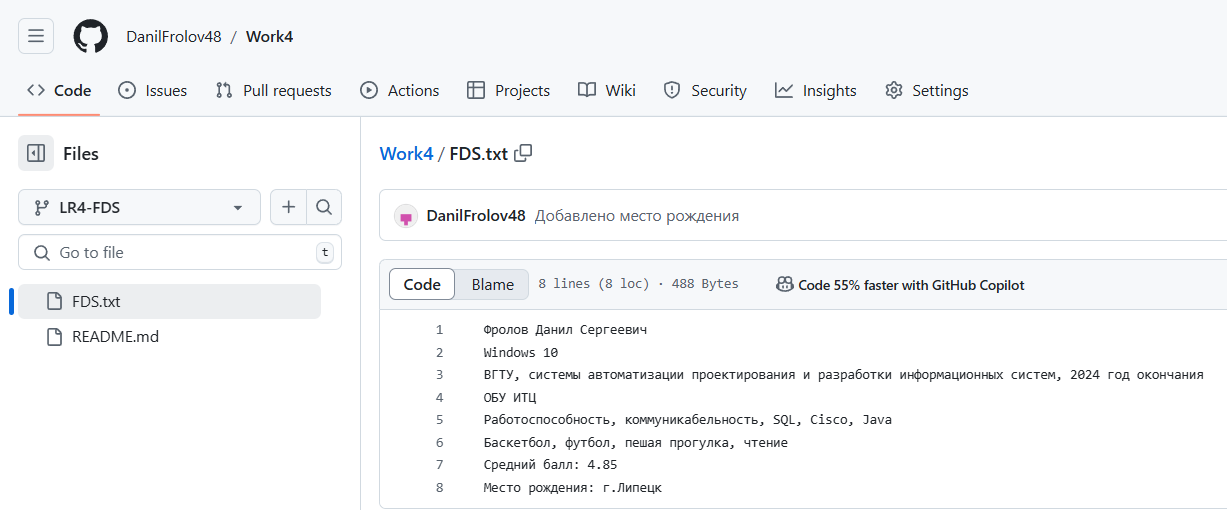


Рисунок 19 – Отображение изменений в файле «FDS.txt»

Далее произведем откат изменения до момента указания информации о месте рождения. Для отката к определенному коммиту используется команда «$ git revert <commit id>». Для того чтобы узнать <commit id> необходимо выполнить команду git log в терминале. Выполнение команды git log представлен на рисунке 20.

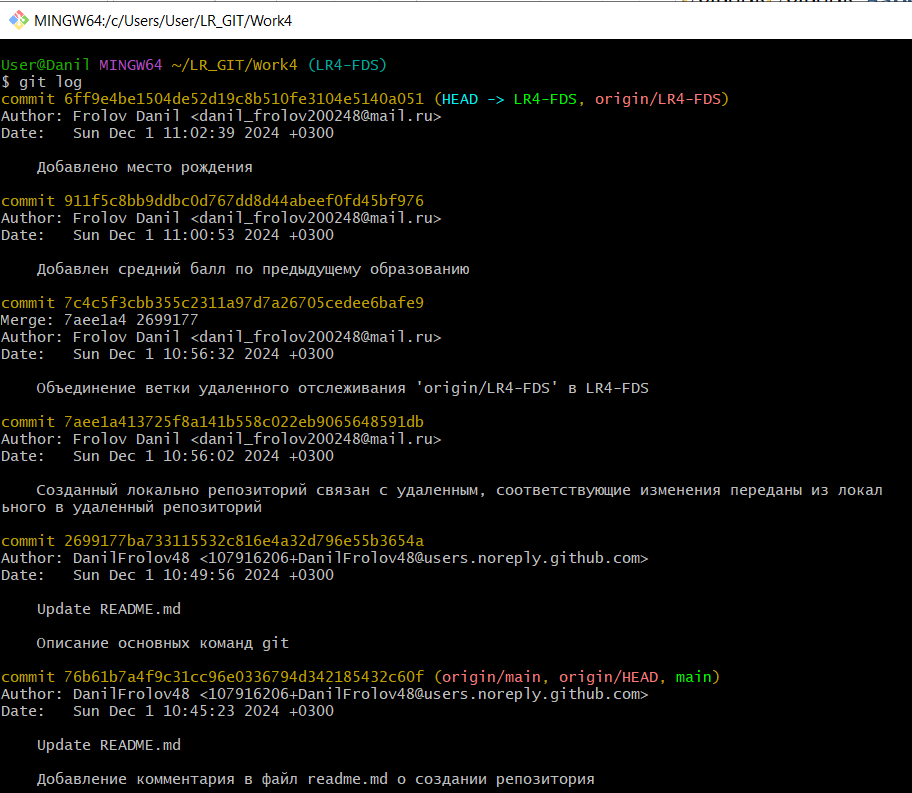


Рисунок 20 – Выполнение команды git log

Выполним команду $ git revert 6ff9e4be1504de52d19c8b510fe3104e5140a051. Далее выполним команды $ git add FDS.txt и $ git push. Результат представлен на рисунках 21-22.

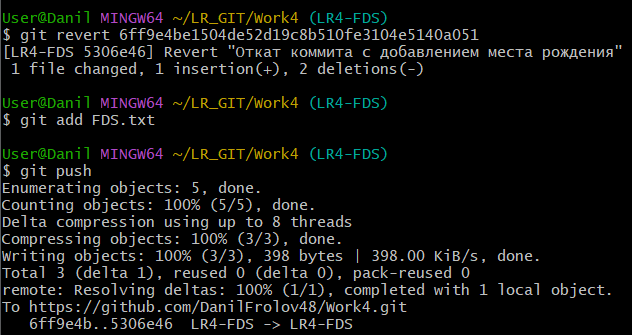


Рисунок 21 – Выполнение отката изменений до предыдущего коммита

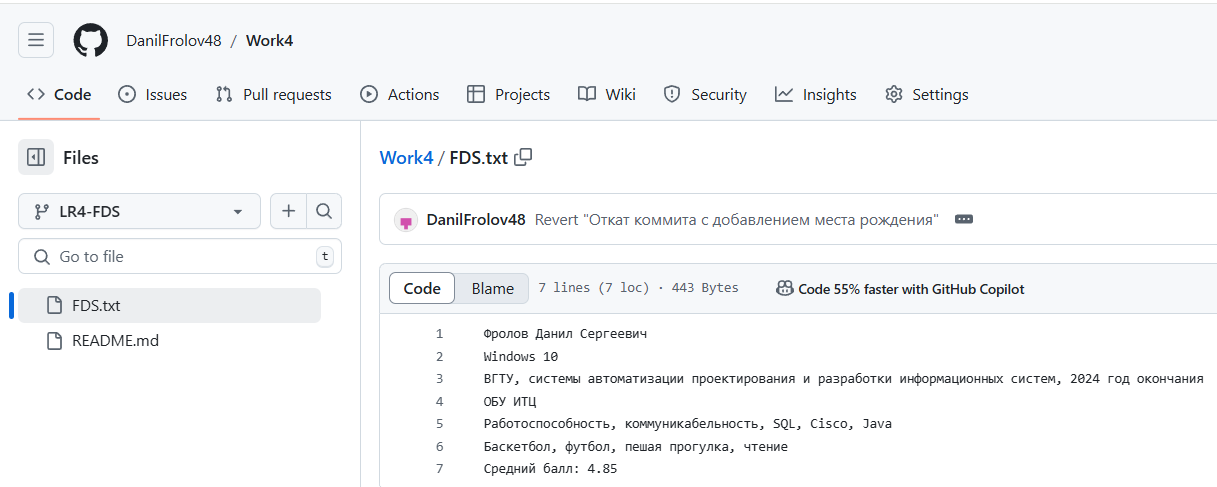


Рисунок 22 – Результат выполнения отката

4. Выводы по результатам лабораторной работы

В результате выполнения данной лабораторной работы было изучено назначение распределенной системы управления версиями, освоен процесс создания репозитория и основы управления версиями файлов

5. Ответы на контрольные вопросы

1. Git – это распределенная система управления версиями. Она позволяет отслеживать любые изменения в файлах, хранить их версии и оперативно возвращаться в любое сохраненное состояние. Git хранит информацию не в виде списка изменений в файлах, а в виде набора снимков – полное отображение того, как выглядит файл в момент сохранения. Это позволяет всегда иметь полную информацию обо всех файлах и быстро восстанавливать любую из предыдущих версий.

2. Команды разработки используют Git для выполнения множества задач, связанных с управлением исходным кодом и координацией работы над проектами. Основные причины, по которым команды разработки используют Git:

- управление версиями кода. Git сохраняет полную историю изменений в кодовой базе, позволяя отслеживать, какие изменения были внесены, когда и кем. В случае ошибок или нежелательных изменений можно легко откатиться к предыдущей стабильной версии кода;

- совместная работа. Разработчики могут работать над разными частями проекта одновременно без конфликтов, используя различные ветки;

- ветвление и экспериментирование. Команды могут создавать отдельные ветки для разработки новых функций, исправления ошибок или проведения экспериментов, не затрагивая основную кодовую базу;

- координация и коммуникация. Git вместе с платформами GitHub, GitLab позволяет разработчикам предлагать изменения через pull-запросы, которые могут рассмотрены и обсуждены командой перед слиянием;

- надежность и безопасность. Каждому разработчику предоставляется полная копия репозитория, что обеспечивает надежность данных и возможность работы в автономном режиме. Благодаря распределенной природе Git, утрата данных минимальна, так как каждый участник обладает полной историей проекта;

- интеграция с другими инструментами. Git легко интегрируется с системами непрерывной интеграции и доставки, автоматизируя процессы сборки, тестирования и деплоя;

- поддержка открытых проектов. Git идеально подходит для проектов с открытым исходным кодом, позволяя сообществу вносить свой вклад.

3. Основные команды и операции, необходимые для работы в Git:

- инициализация репозитория (git init). Создает новый локальный Git-репозиторий в текущей директории;

- клонирование репозитория (git clone <URL>). Клонирует существующий удаленный репозиторий по указанному URL в локальную директорию;

- проверка текущего состояния репозитория (git status). Показывает изменения, добавленные в индекс, неотслеживаемые файлы и текущую ветку;

- добавление файлов в индекс (git add <имя\_файла>). Добавляет изменения указанного файла в индекс (подготовка к коммиту);

- фиксация изменений (git commit –m «сообщение»). Создает новый коммит с подготовленными изменениями и описанием;

- просмотр коммитов и логов. Команда git log показывает подробную историю коммитов в репозитории. Команда git log --oneline отображает историю коммитов в сжатом формате, по одному коммиту на строку. Команда git diff показывает различия между рабочими файлами и индексом или между коммитами;

- работа с ветками. Команда git branch отображает список всех локальных веток в репозитории. Команда git branch <имя\_ветки> создает новую ветку с указанным именем. Команда git checkout <ветка> переключается на указанную ветку. Команда git checkout –b <ветка> создает новую ветку и сразу переключается на нее. Команда git merge <ветка> сливает указанную ветку с текущей. Команда git branch –d <ветка> удаляет указанную локальную ветку.

- отправка изменений в удаленный репозиторий (git push origin <имя\_ветки>). Загружает изменения на удаленный сервер;

- получение изменений из удаленного репозитория (git pull). Загружает изменения с удаленного репозитория и выполняет слияние с текущей веткой;

- скачивание изменений без слияния (git fetch). Получает изменения из удаленного репозитория, но не сливает их с текущей веткой$

- сброс изменений. Чтобы сбросить коммит, но оставить изменения в рабочей директории необходимо выполнить git reset --soft HEAD ~1. Чтобы сбросить изменения и удалить их из рабочей директории необходимо выполнить git reset --hard HEAD ~1.

4. Инициализация репозитория – это процесс создания нового локального репозитория Git в выбранной директории. После инициализации в папке проекта создается скрытая папка .git, которая содержит все необходимые данные для работы Git (метаданные, настройки, ветки и т.д.). Для инициализации репозитория используется команда git init.