|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных технологий (МОСИТ)

|  |
| --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1** |
| **по дисциплине** |
| **«Технология разработки программных приложений»**  **Тема: «Системы контроля версий»** |

|  |  |
| --- | --- |
| Выполнил студент группы ИКБО-20-23 | Кузнецов Л. А. |
| Принял | Чернов Е. А. |

|  |  |  |
| --- | --- | --- |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2025

# оглавление

[ЗАДАНИЕ………………………………………………………………………..4](#a0)

[Часть 1. Как начать работу с git’ом?...................................................................4](#a1)

[Шаг 1. Устанавливаем Git на локальную машину 4](#a2)

[Шаг 2. Настройка Git 4](#a3)

[Шаг 3. Начало работы – создание папки и файлов 5](#a4)

[Шаг 4. Создание репозитория 6](#a5)

[Шаг 5. Добавление файла в репозиторий и добавление первого коммита……………………………………………………………………………6](#a6)

[Шаг 6. Индексация изменений 6](#a7)

[Шаг 6\*. Индексация и коммит (не обязательный) – для ознакомления .6](#a8)

[Шаг 7. Коммиты нескольких изменений 7](#a9)

[Шаг 8. Просмотр истории коммитов 8](#a10)

[Шаг 9. Получение старых версий 9](#a11)

[Шаг 10. Отмена локальных изменений (до индексации) 11](#a12)

[Шаг 11. Отмена локальных изменений (после индексации и до коммита)………………………………………………………………………..12](#a13)

[Шаг 12. Отмена коммита 13](#a14)

[Часть 2. Управление репозиториями 14](#a15)

[Шаг 1. Создание аккаунта на GitHub 14](#a16)

[Шаг 2. Создание SSH-ключа для авторизации 14](#a17)

[Шаг 3. Создание нового репозитория для своего проекта 16](#a18)

[Шаг 4. Связываем локальный и удаленный репозитории 17](#a19)

[Шаг 5. Создание веток и переключение между ними 18](#a20)

[Шаг 6. Слияние веток 18](#a21)

[Часть 3. Ветвление и оформление кода 19](#a22)

[Шаг 1. Форк репозитория 19](#a23)

[Шаг 2. Клонирование репозитория на локальную машину 20](#a24)

[Шаг 3. Создание двух веток в master'е 21](#a25)

[Шаг 4. 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла ..21](#a26)

[Шаг 5. Слияние ветки branch1 в ветку branch2 с разрешением конфликтов……………………………………………………………………….22](#a27)

[Шаг 6. Выгрузка всех изменений во всех ветках в удаленный репозиторий……………………………………………………………………....22](#a28)

[Шаг 7. Выполнение 3-х коммитов в branch1 23](#a29)

[Шаг 8. Повторное клонирование репозитория в другую директорию .23](#a30)

[Шаг 9. Выполнение 3-х коммитов в ветку branch1 24](#a31)

[Шаг 10. Выгрузка всех изменений в удаленный репозиторий 24](#a32)

[Шаг 11. Выгрузка изменения из старого репозитория с опцией --force.24](#a33)

[Шаг 12. Получение всех изменения в новом репозитории 24](#a34)

[Часть 4. Индивидуальное задание №1……………………………………….24](#a35)

[Часть 5. Индивидуальное задание №2……………………………………….28](#a36)

[Часть 6.Ответы на вопросы…………………………………………………..35](#a37)

[Вывод…………………………………………………………………………...36](#a38)

# задание

**Цель работы:** получить навыки по работе с командной строкой и git’ом.

**Часть 1. Основные команды Git**

**Шаг 1. Устанавливаем Git на локальную машину**

Проведём работу по заданию.

Установили Git на Windows 10 (Рисунок 1).

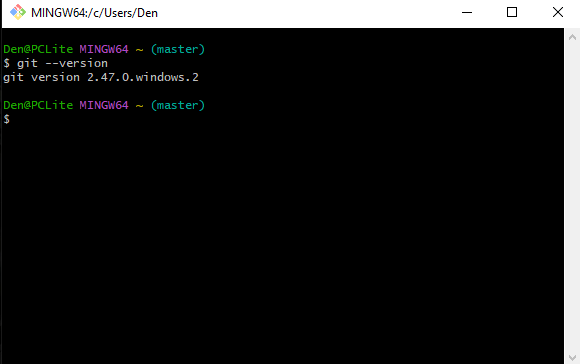


Рисунок 1 – Открытая консоль Git на Windows

**Шаг 2. Настройка Git**

Настроим Git в соответствии с требованиями (Рисунки 2-3).

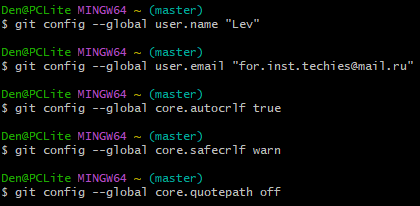


Рисунок 2 – Ввод команд настройки Git-а





Рисунок 3 – Результат настройки на Git

**Шаг 3. Начало работы – создание папки и файлов**

На Рисунке 4 работает по заголовку. На Рисунке 5 изображён созданный файл.

****

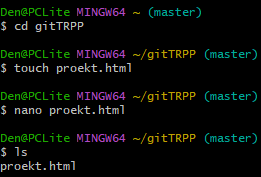


Рисунок 4 – Работа с папками

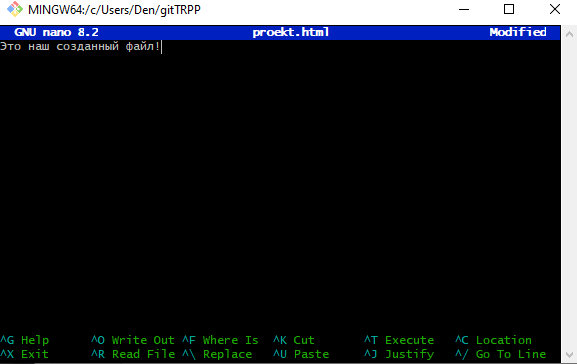


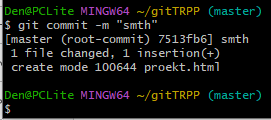
Рисунок 5 – Состав файла proekt.html

**Шаг 4. Создание репозитория**



Рисунок 6 – Создание репозитория

**Шаг 5. Добавление файла в репозиторий и добавление первого коммита**

****

****

Рисунок 7 – Добавление файла в репозиторий и добавление первого коммита

**Шаг 6. Индексация изменений**

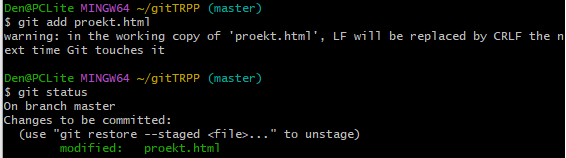
****

Рисунок 8 – Фиксация изменений файла в Git

**Шаг 6\*. Индексация и коммит (не обязательный) – для ознакомления**

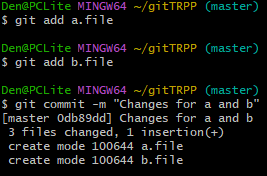
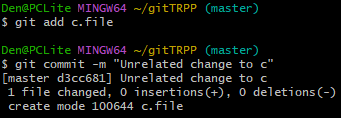
**** ****

Рисунок 9 – Индексация и коммит

**Шаг 7. Коммиты нескольких изменений**

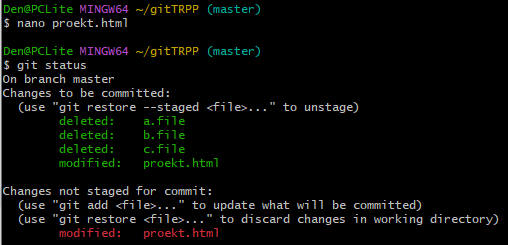
****

Рисунок 10 – 1 часть коммита нескольких изменений

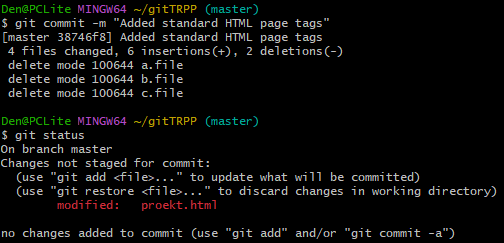
****

Рисунок 11 – 2 часть коммита нескольких изменений

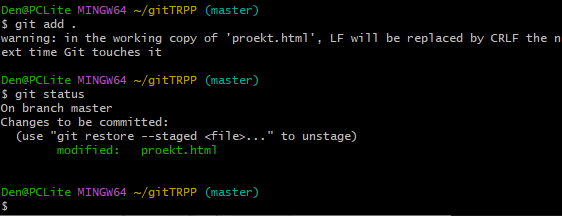
****

Рисунок 12 – 3 часть коммита нескольких изменений

**Шаг 8. Просмотр истории коммитов**

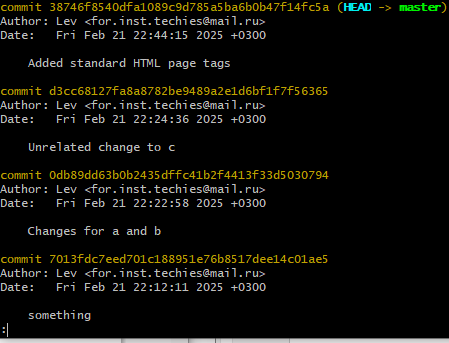


Рисунок 13 – 1 часть просмотра истории коммитов

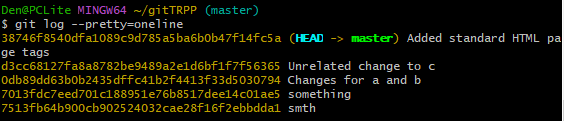
****

Рисунок 14 – 2 часть просмотра истории коммитов

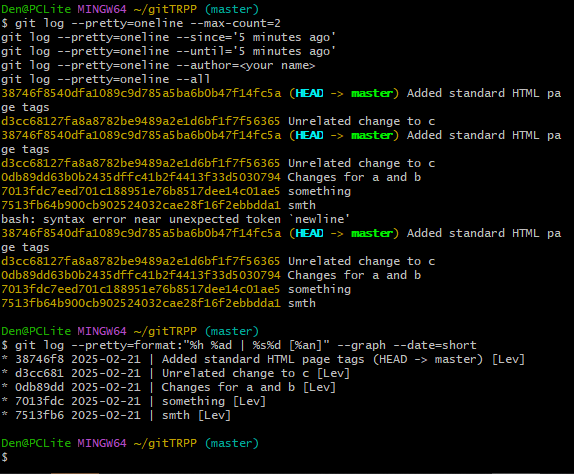
****

Рисунок 15 – 3 часть просмотра истории коммитов

**Шаг 9. Получение старых версий**

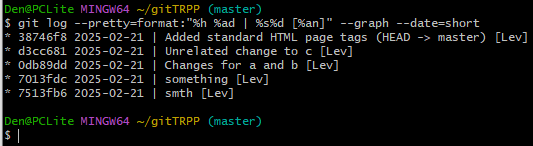
****

Рисунок 16 – Получение всех коммитов

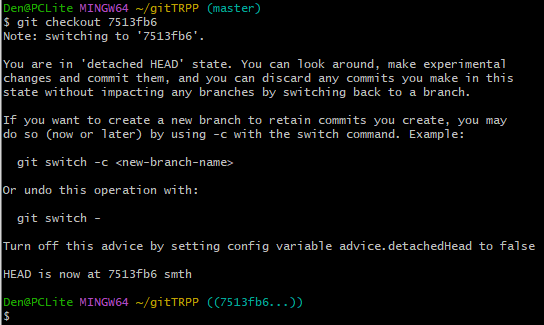
****

Рисунок 17 – Изменение файлов на старые версии

****

Рисунок 18 – Проверка на изменения в файлах

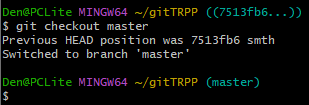
****

Рисунок 19 – Смена версии на новую

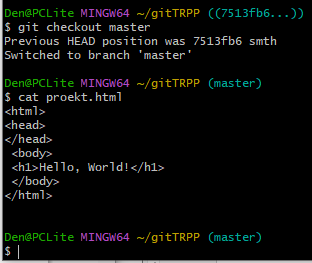
****

Рисунок 20 – Проверка возвращения к новейшей

**Шаг 10. Отмена локальных изменений (до индексации)**

****

Рисунок 21 – Проверка текущей версии

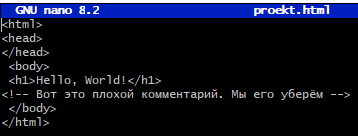
****

Рисунок 22 – Изменяем файл

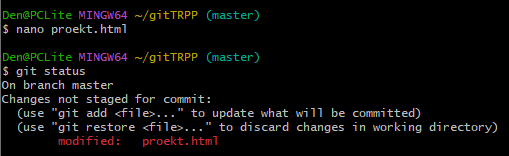
****

Рисунок 23 - Проверка наличия изменений

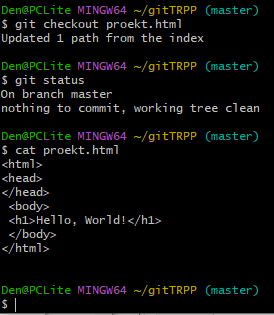
****

Рисунок 24 – Отмена изменений

**Шаг 11. Отмена локальных изменений (после индексации и до коммита)**

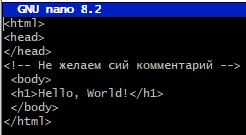
****

Рисунок 25 – Делаем изменения в файле

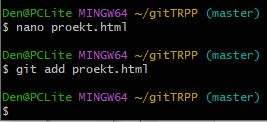
****

Рисунок 26 – Добавление коммита в рассмотрение изменений git-а

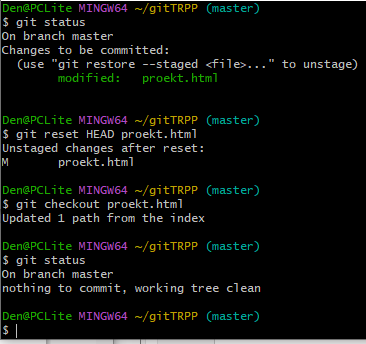
****

Рисунок 27 – Отмена изменения после индексации и до коммита

**Шаг 12. Отмена коммита**

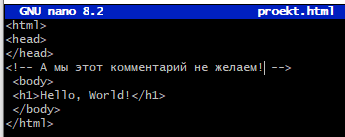
****

Рисунок 28 – Изменение в файле proekt.html

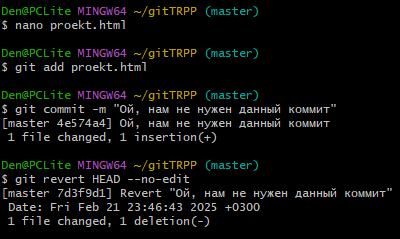
****

Рисунок 29 – Отмена коммита

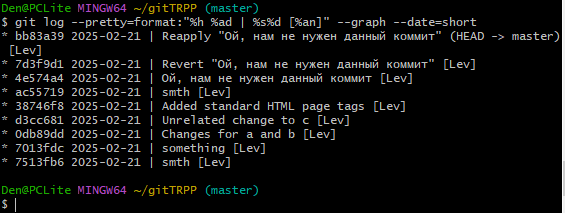
****

Рисунок 30 – Проверка истории коммитов

**Часть 2. Управление репозиториями**

**Шаг 1. Создание аккаунта на GitHub**

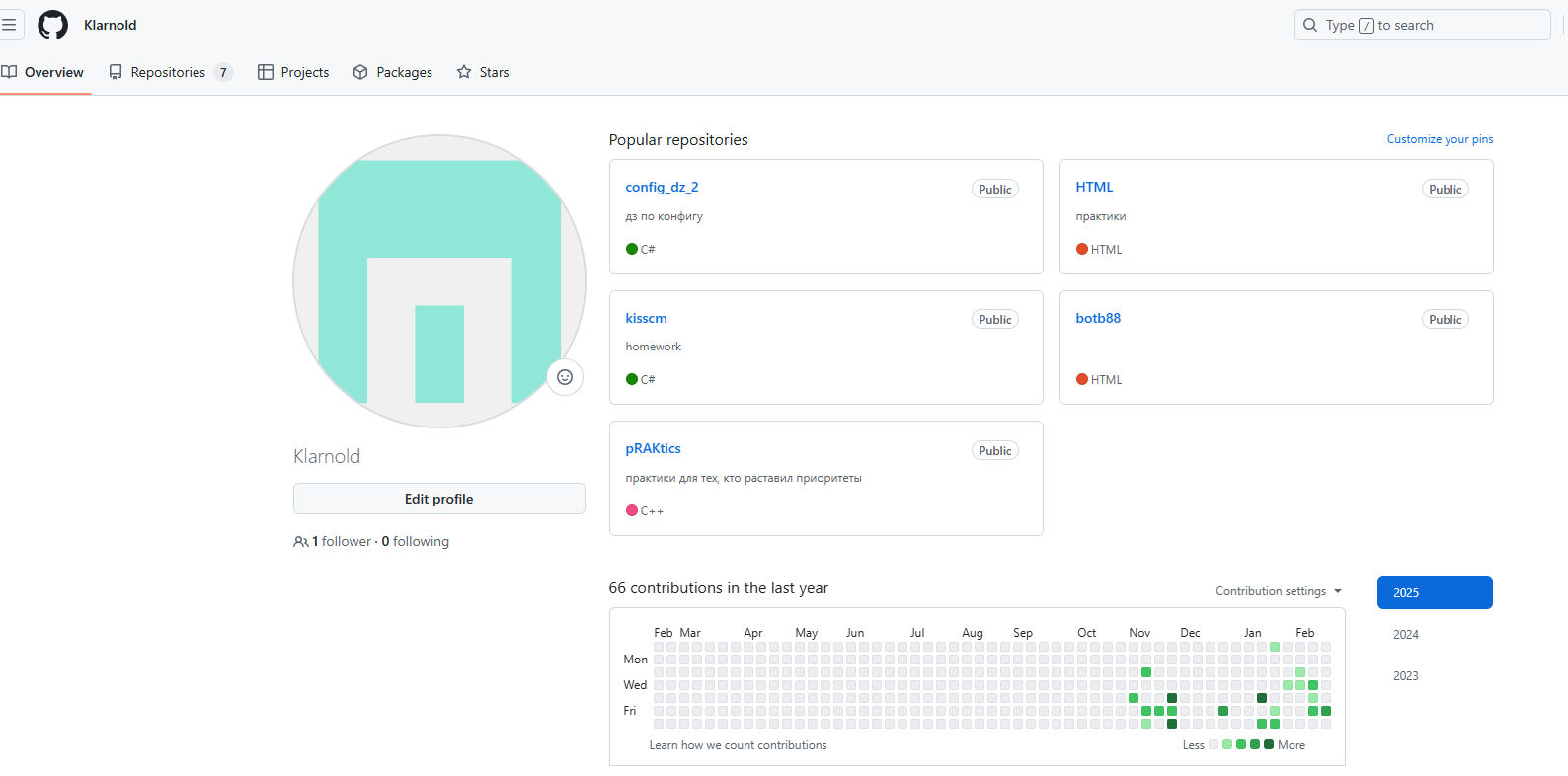
****

Рисунок 31 – Профиль аккаунта на GitHub-е

**Шаг 2. Создание SSH-ключа для авторизации**

****

Рисунок 32 – Генерация SSH ключа

****

Рисунок 33 – Получение SSH ключа

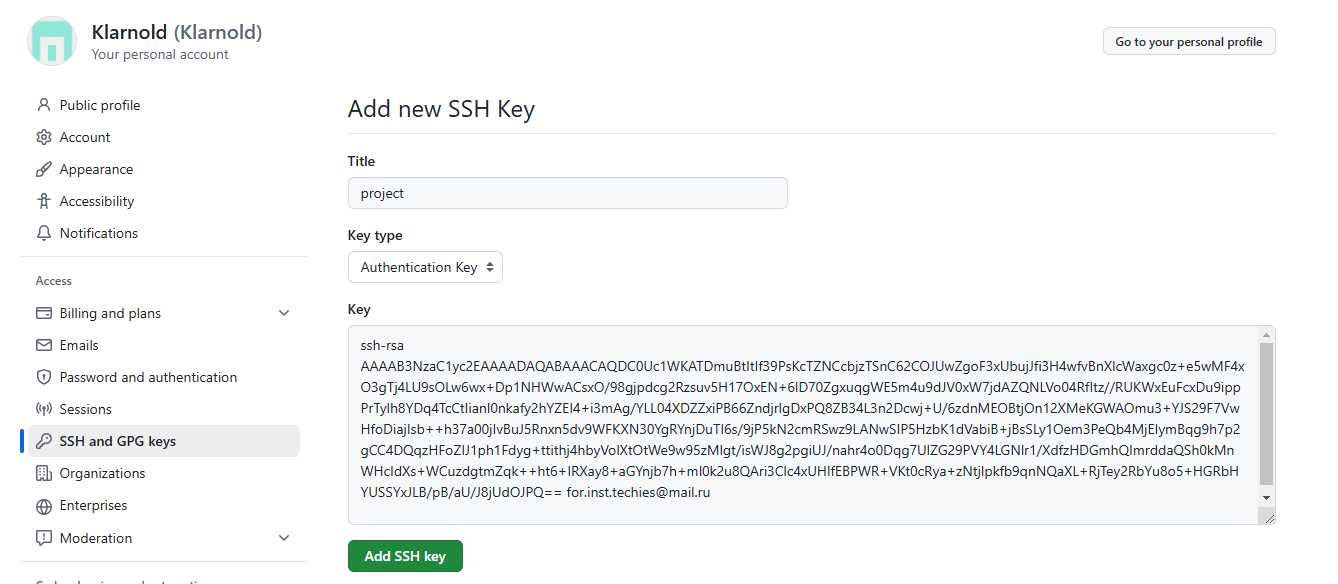
****

Рисунок 34 – 1 часть добавления SSH ключа на GitHub

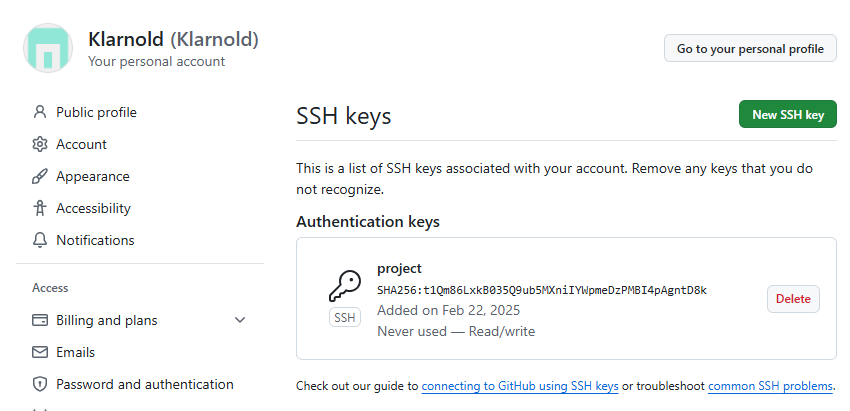
****

Рисунок 35 – 2 часть добавления SSH ключа на GitHub

**Шаг 3. Создание нового репозитория для своего проекта**

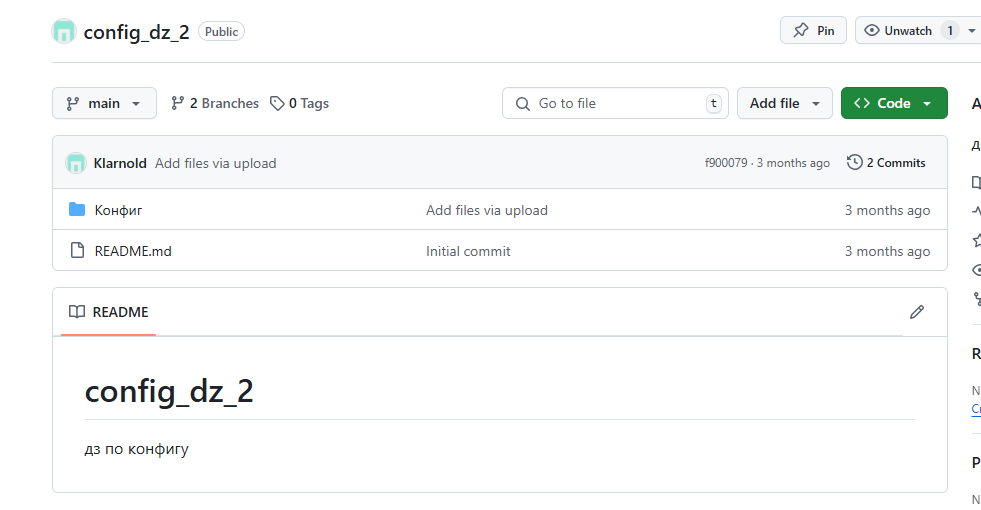
****

Рисунок 36 – Создание нового репозитория

**Шаг 4. Связываем локальный и удаленный репозитории**

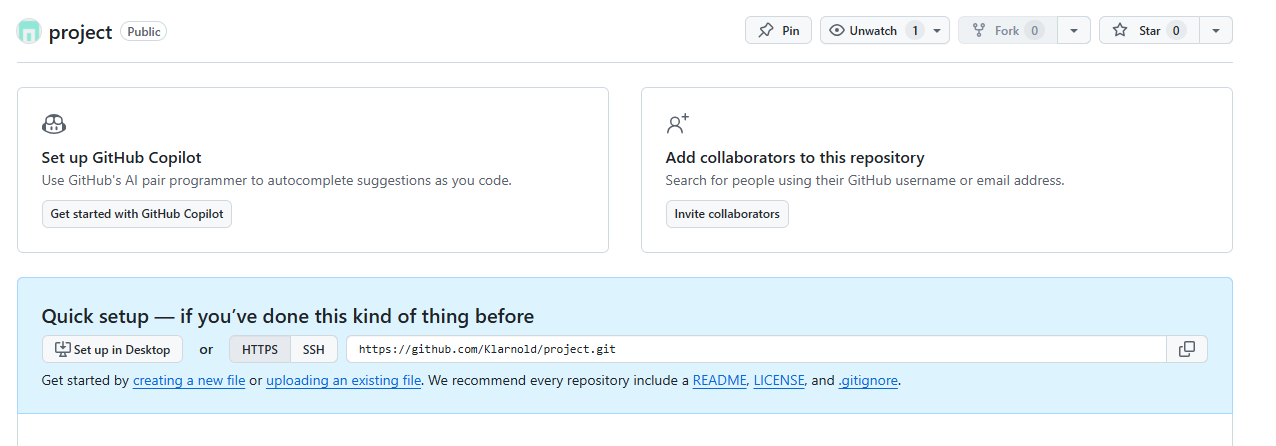
****

Рисунок 37 – Создание нового репозитория

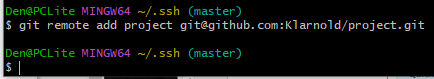
****

Рисунок 38 – Связь локального и удалённого репозиториев

**Шаг 5. Создание веток и переключение между ними**

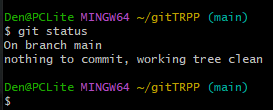
****

Рисунок 39 – Проверка основной ветки

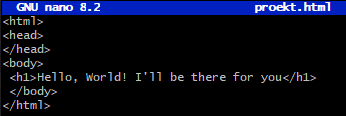
****

Рисунок 40 – Изменение файла

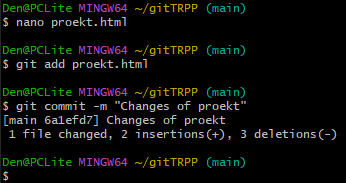
****

Рисунок 41 – Индексация и коммит изменений

**Шаг 6. Слияние веток**

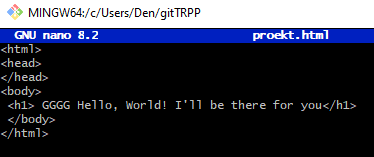
****

Рисунок 42 – Изменение исходного файла

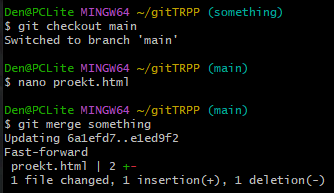
****

Рисунок 43 – Слияние веток

**Часть 3. Ветвление и оформление кода**

**Шаг 1. Форк репозитория**

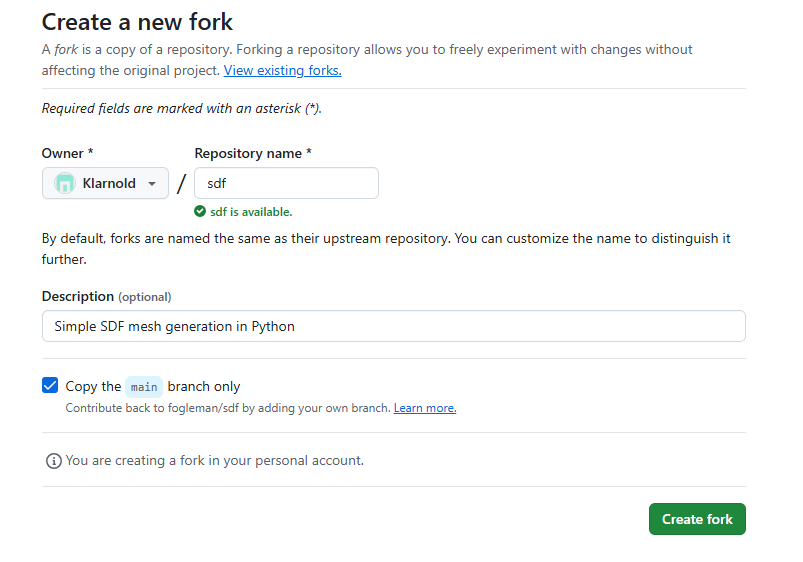
****

Рисунок 44 – Форк репозитория

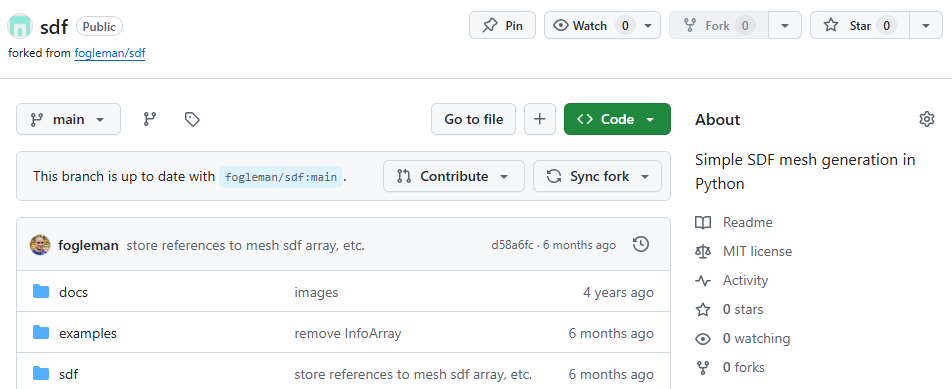
****

Рисунок 45 – Репозиторий после форка

**Шаг 2. Клонирование репозитория на локальную машину**

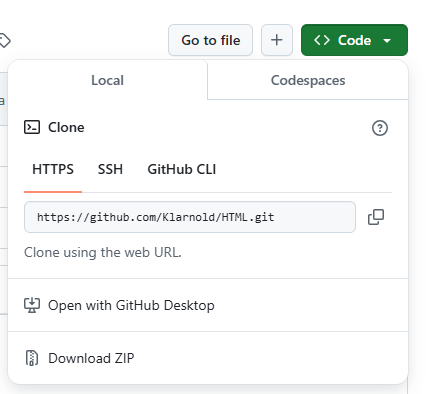
****

Рисунок 46 – Ссылка на клонируемы репозиторий

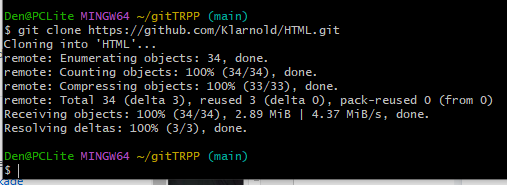
****

Рисунок 47 – Производим процесс клонирования

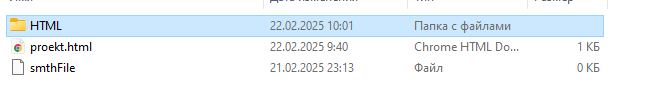
****

Рисунок 48 – Результат клонирования

**Шаг 3. Создание двух веток в master'е**

****

Рисунок 49 - Создание двух веток в master'е

**Шаг 4. 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла**

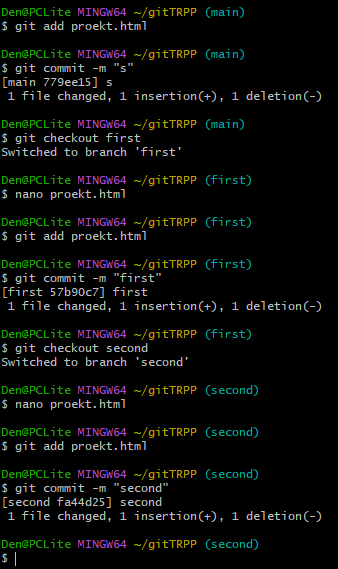
****

Рисунок 50 - 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла

**Шаг 5. Слияние ветки branch1 в ветку branch2 с разрешением конфликтов**

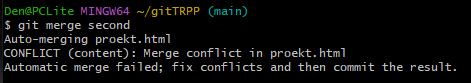


Рисунок 51 – Возникновение конфликта версий

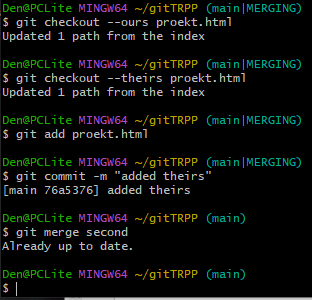


Рисунок 52 – Решение конфликта выбором версии (остаётся second)

**Шаг 6. Выгрузка всех изменений во всех ветках в удаленный репозиторий**

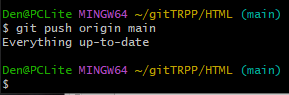
****

Рисунок 53 – Отправка изменений в удалённый репозиторий

**Шаг 7. Выполнение 3-х коммитов в branch1 (Часть 1. Шаг 5)**

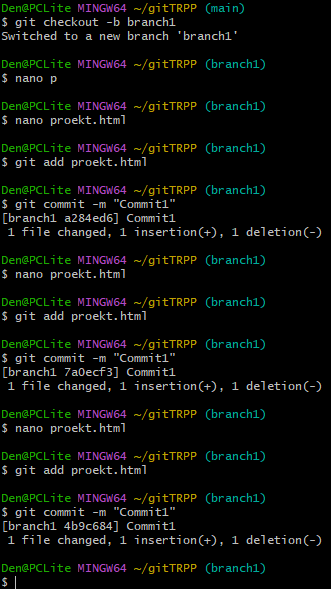
****

Рисунок 54 - Выполнение 3-х коммитов в branch1

**Шаг 8. Повторное клонирование репозитория в другую директорию**

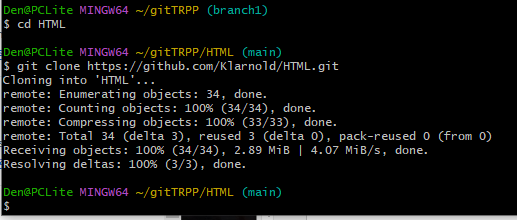
****

Рисунок 55 –Повторное клонирование репозитория в другую директорию

**Шаг 9. Выполнение 3-х коммитов в ветку branch1**

Аналогично шагу 7.

**Шаг 10. Выгрузка всех изменений в удаленный репозиторий**

****

Рисунок 56 - Выгрузка всех изменений в удаленный репозиторий

**Шаг 11. Выгрузка изменения из старого репозитория с опцией –force**

****

Рисунок 57 - Выгрузка изменения из старого репозитория с опцией –force

**Шаг 12. Получение всех изменения в новом репозитории**

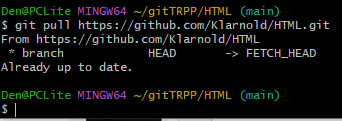
****

Рисунок 58 - Получение всех изменений в новом репозитории

**Часть 4. Индивидуальное задание №1**

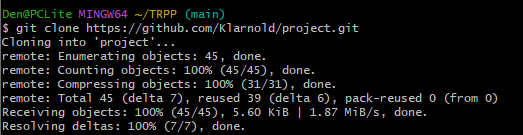
****

Рисунок 59 – Клонируем удалённый репозиторий

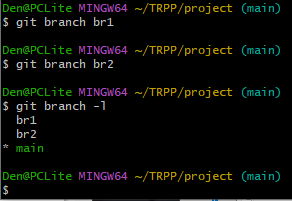
****

Рисунок 60 – Создаём две ветви

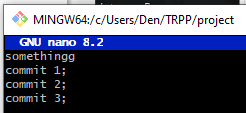
****

Рисунок 61 – Итог трёх коммитов

****

Рисунок 62 – Просмотр версий br1

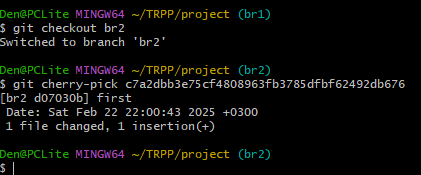
****

Рисунок 63 – Переводим первый коммит с br1 на br2

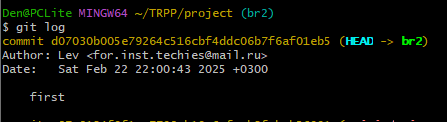
****

Рисунок 64 – Проверка версий br2

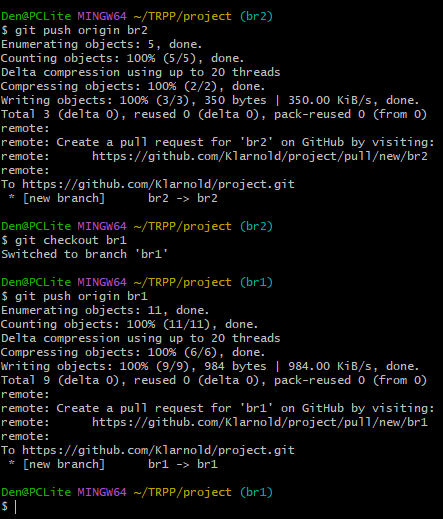
****

Рисунок 65 – Выгрузка изменений в удалённый репозиторий

****

Рисунок 66 – Файл smth в br1

****

Рисунок 67 – Файл smth в br2

****

Рисунок 68 – Файл smth в main

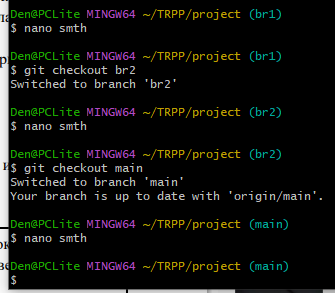
****

Рисунок 69 – Вывод содержимого файла smth в различных ветвях

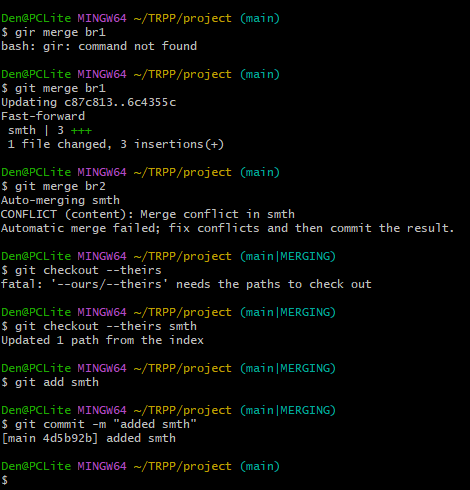
****

Рисунок 70 – Слияние веток br1и br2 с main

**Часть 5. Индивидуальное задание №2**

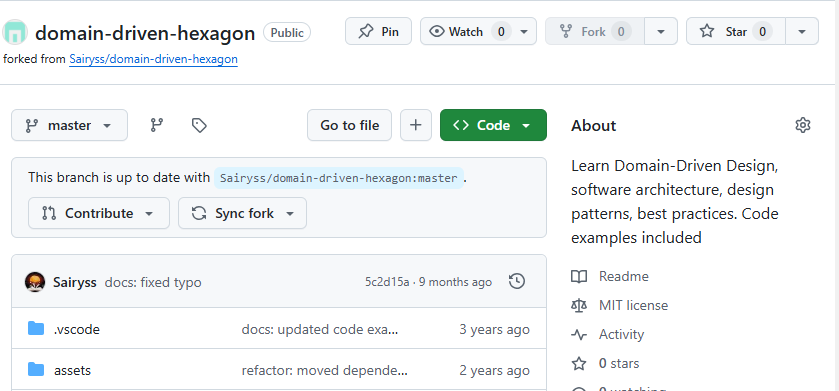
****

Рисунок 71 – Форк репозитория согласно варианту

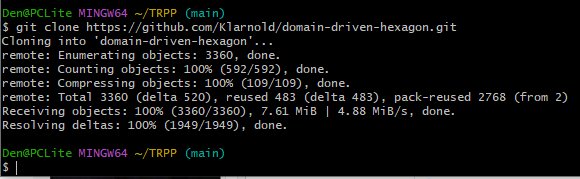
****

Рисунок 72 – Клонирование форк-а на локальную машину

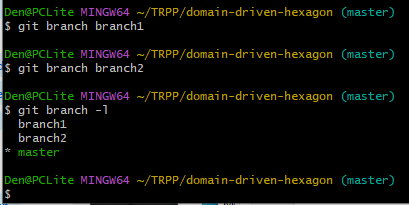
****

Рисунок 73 – Создание веток branch1 и branch2

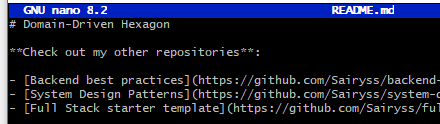
****

Рисунок 74 – Изначальный вид файла README.md

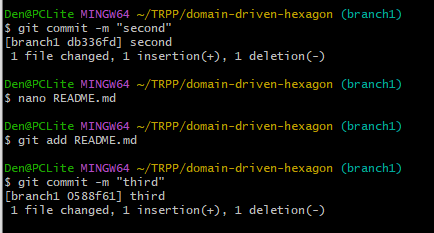
****

Рисунок 75 – Осуществление коммитов

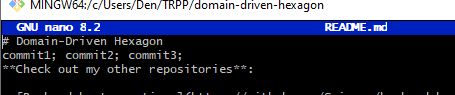
****

Рисунок 76 – Результат 3 коммитов для файла README.md в ветке branch1

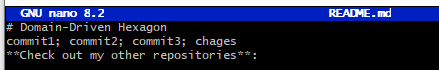
****

Рисунок 77 – Результат 3 коммитов для файла README.md в ветке branch2

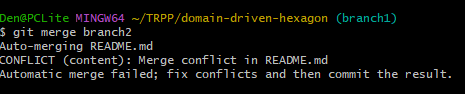
****

Рисунок 78 – Возникновение конфликта

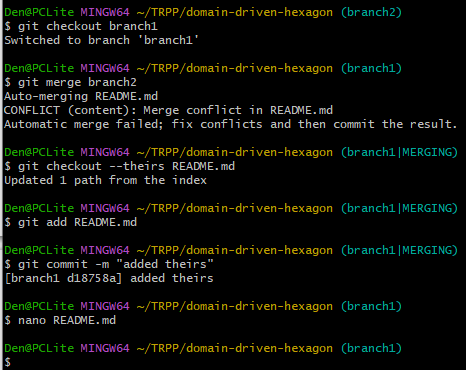
****

Рисунок 79 – Решение конфликта слияния

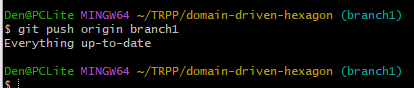
****

Рисунок 80 – Выгрузка изменений в ветке branch1 в удалённый репозиторий

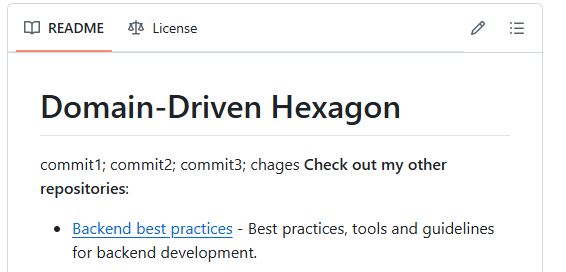
****

Рисунок 81 – Изменения в ветке branch2

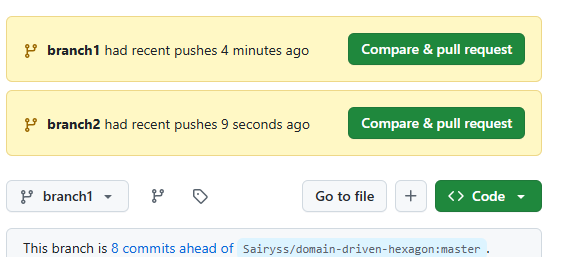
****

Рисунок 82 – Наличие веток branch1 и branch2 в удалённом репозитории

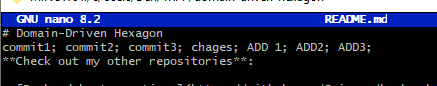
****

Рисунок 83 – Результат 3 коммитов для файла README.md в ветке branch1

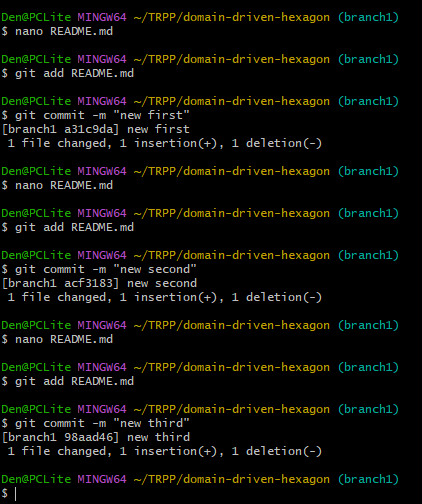
****

Рисунок 84 – Осуществление коммитов в branch1

****

Рисунок 85 – Клонирование удалённого репозитория в новую папку

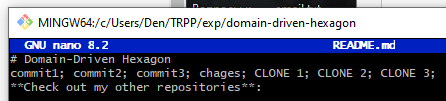
****

Рисунок 86 – Результат коммитов в branch1 для файла README.md

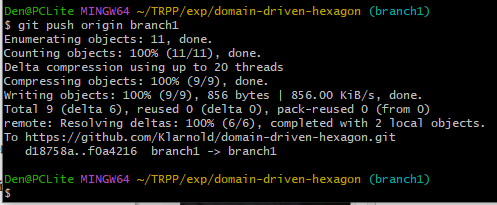
****

Рисунок 87 – Выгрузка изменений с нового клона в удалённый репозиторий

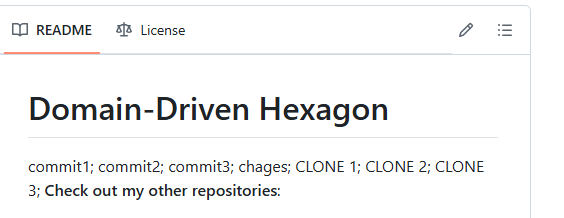
****

Рисунок 88 – Отображение изменений в удалённом репозитории

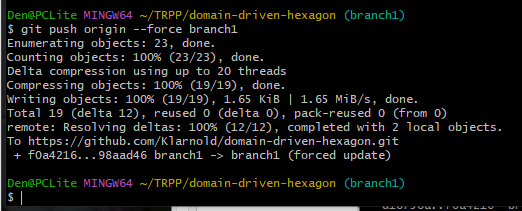
****

Рисунок 89 – Обновление удалённого репозитория при помощи --force

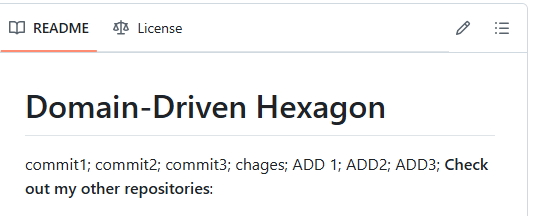
****

Рисунок 90 - Отображение изменений в удалённом репозитории

**Часть 6. Ответы на вопросы**

4. Что такое коммит?

Коммит – это фиксация изменений, произведённых в текущем репозитории.

5. Что такое ветка в репозитории Git?

Ветка в Git – это набор изменений, сохраняющий хронологический порядок. Можно сказать, что изначально репозиторий состоит из одной ветки. Ветки можно сливать друг с другом, а также переходить от коммита к коммита.

7. Что такое слияние двух веток?

Слияние двух веток – это перенос изменений с одной ветки на другую. Если мы говорим о слиянии branch1 с branch2, то в таком случае все изменения с branch1 перейдут в branch2.

8. Что такое конфликт в Git? Как его решить и почему они бывают?

Конфликт в Git – это когда при слияни двух веток изменения одной ветки противоречат другой – то есть разные изменения в одинаковых файлах и система не может решить, какие изменения стоит записать. Решить это проблему можно при помощи выбора вручную нужней версии или при устранении нежелаемого коммита в самом файле с удалением разметкой, обозначающей два разных коммита.

9. Как отменить слияние веток, если произошел конфликт?

Отменить слияние можно при помощи команды git merge –abort.

13. Что делает команда git log?

Команда git log выводит на экран коммиты текущей ветви в хронологическом порядке (новейшие коммиты находятся сверху) с отображением положения ветвей на определённых коммитах.

25. Как отменить действие команды "git add" на файл?

Отменить действие команды git add можно при помощи команд git reset <наименование файла> (git reset – отменяет действие команды для всех файлов).

32. Что такое GitHub?

GitHub - веб-сервис для хостинга IT-проектов и их совместной разработки. По сути, это визуальный интерфейс для работы с системой контроля версий Git. Также GitHub предоставляет глобальное взаимодействие между репозиториями, позволяя пользователям комментировать их и обращаться к ним.

**Вывод**

В ходе я ознакомился с такой прекрасным инструментом как Git, который позволяет вести контроль и сохранение старых версий репозиториев, что значительно облегчает работу программиста. Кроме ознакомления, я на практике закрепил полученные знания и обрёл навыки, которые непременно пригодятся мне в дальнейшей карьере программиста.