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

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**Отчет по практической работе №1**

по дисциплине

«**Технология разработки программных приложений**»

**Тема: «Системы контроля версий»**

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-36-22 | Утенков Юрий Юрьевич |
| **Проверил:** | ассистент Петрова А.А. |

МОСКВА 2024 г.

**О Г Л А В Л Е Н И Е**

[Практическая работа № 1 3](#_Toc159271450)

[Цель работы: 3](#_Toc159271451)

[Выполнение работы 3](#_Toc159271452)

[Часть 1. 6](#_Toc159271453)

[Часть 2. 18](#_Toc159271454)

[Часть 3. 30](#_Toc159271455)

[Контрольные вопросы. 43](#_Toc159271456)

[Выводы 43](#_Toc159271457)

# Практическая работа № 1

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

Выполнение работы

**Задание:**

**Часть 1. Основные команды Git**

1. Установите и настройте клиент git на своей рабочей станции.

2. Создайте локальный репозиторий и добавьте в него несколько файлов.

3. Внесите изменения в один из файлов.

4. Проиндексируйте изменения и проверьте состояние.

5. Сделайте коммит того, что было проиндексировано в репозиторий. Добавьте к коммиту комментарий.

6. Измените еще один файл. Добавьте это изменение в индекс git. Измените файл еще раз. Проверьте состояние и произведите коммит проиндексированного изменения. Теперь добавьте второе изменение в индекс, а затем проверьте состояние с помощью команды git status. Сделайте коммит второго изменения.

7. Просмотрите историю коммитов с помощью команды git log. Ознакомьтесь с параметрами команды и используйте некоторые из них для различного формата отображения истории коммитов.

8. Верните рабочий каталог к одному из предыдущих состояний.

9. Изучите, как создавать теги для коммитов для использования в будущем.

10. Отмените некоторые изменения в рабочем каталоге (до и после индексирования).

11. Отмените один из коммитов в локальном репозитории.

**Часть 2. Системы управления репозиториями**

1. Создайте аккаунт на GitHub (у кого нет)

2. Создайте репозиторий на GitHub и на локальной машине, согласно выбранной теме проекта,

3. Создайте несколько файлов на локальной машине при помощи консоли,

4. Создайте SSH-ключ для авторизации,

5. Свяжите репозиторий локальной машины с репозиторием на GitHub при помощи консоли

6. Создайте новую ветку в репозитории с помощью команды, произведите в ней какие-нибудь изменения, а после слейте с веткой master

7. Выполните цепочку действий в репозитории, согласно вариантам.

|  |  |
| --- | --- |
| № Варианта | Задание |
| 5 | 1) Клонируйте непустой удаленный репозиторий на локальную машину  2) Создайте новую ветку и выведите список всех веток  3) Произведите 3 коммита в новой ветке в разные файлы  4) Выгрузите изменения в удаленный репозиторий  5) Произведите revert предпоследнего коммита в новой ветке  6) Выведите в консоли различия между веткой master и новой веткой  7) Слейте новую ветку с master при помощи merge |

**Часть 3. Работа с ветвлением и оформление кода**

1. Сделайте форк репозитория в соответствии с вашим вариантом

2. Склонируйте его на локальную машину

3. Создайте две ветки branch1 и branch2 от последнего коммита в master'е

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

5. Выполните слияние ветки branch1 в ветку branch2, разрешив конфликты при этом

6. Выгрузите все изменения во всех ветках в удаленный репозиторий

7. Проведите еще 3 коммита в ветку branch1

8. Склонируйте репозиторий еще раз в другую директорию

9. В новом клоне репозитории сделайте 3 коммита в ветку branch1

10. Выгрузите все изменения из нового репозитория в удаленный репозиторий

11. Вернитесь в старый клон с репозиторием, выгрузите изменения с опцией --force

12. Получите все изменения в новом репозитории.

|  |  |
| --- | --- |
| № варианта | Репозиторий |
| 8 | https://github.com/Sairyss/domain-driven-hexagon |

**Ход выполнения работы:**

Часть 1.

Подготовка к работе с Git (рисунок 1).

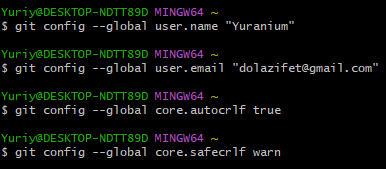


Рисунок 1 – Подготовка к работе с Git

Создание Git проекта (рисунок 2).

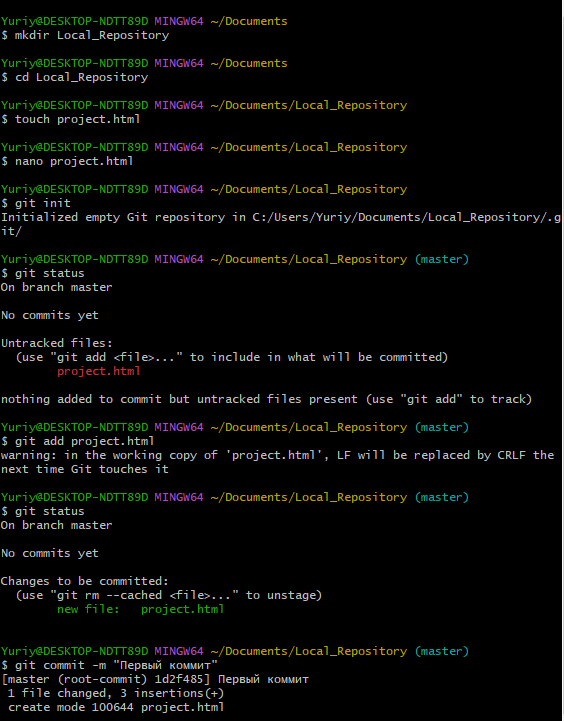


Рисунок 2 – Создание проекта

Проверка статуса Git репозитория (рисунок 3).

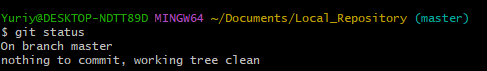


Рисунок 3 – Проверка статуса репозитория

Добавление изменений в Git репозиторий (рисунок 4).

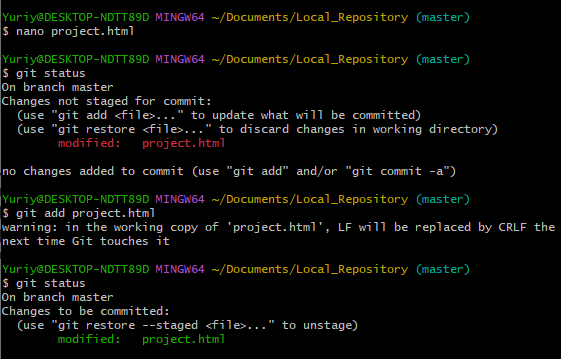


Рисунок 4 – Добавление изменений

Инсценирование изменений в Git репозитории (рисунок 5).

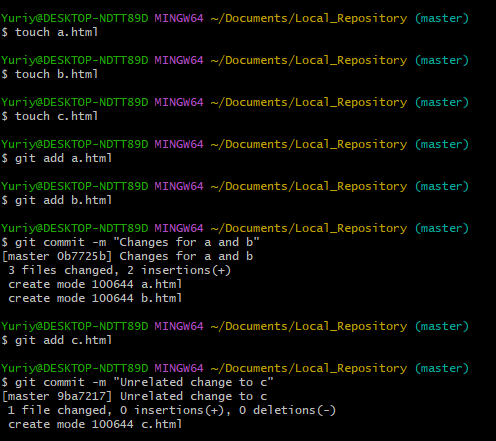


Рисунок 5 – Инсценирование изменений

Коммит изменений в редакторе при помощи команды git commit (рисунок 6).

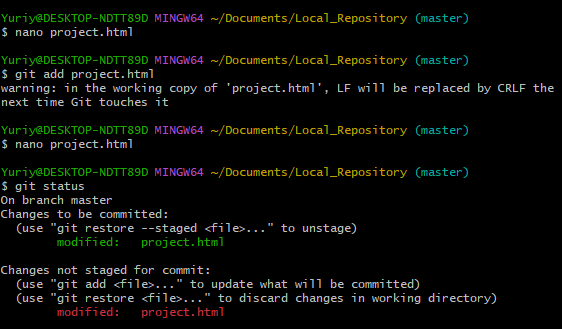


Рисунок 6 – Редактор коммита

Результат после закрытия редактора коммита (рисунок 7).

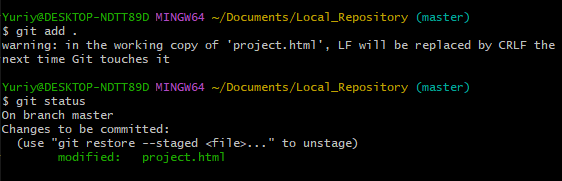


Рисунок 7 – Вывод после закрытия коммита

Git работает с изменениями, а не с самими файлами (рисунок 8).

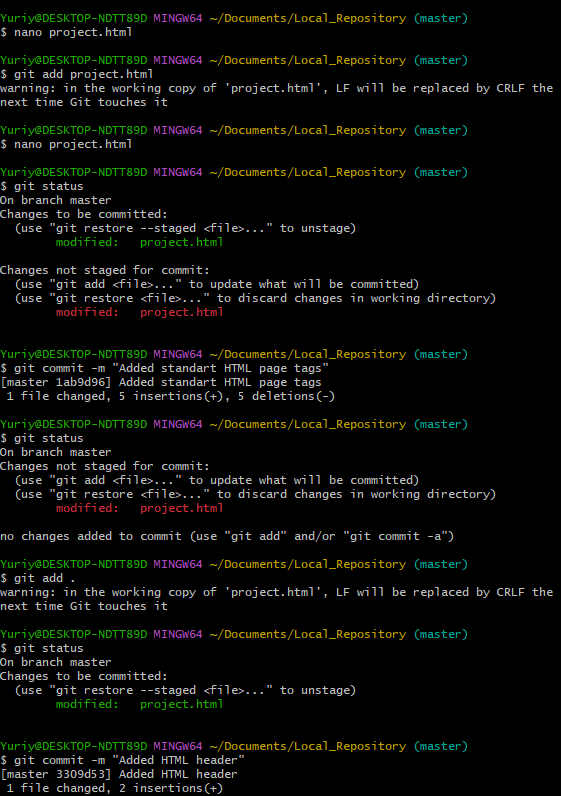


Рисунок 8 – Изменения файлов не происходит

Просмотр списка изменений git проекта (рисунок 9).

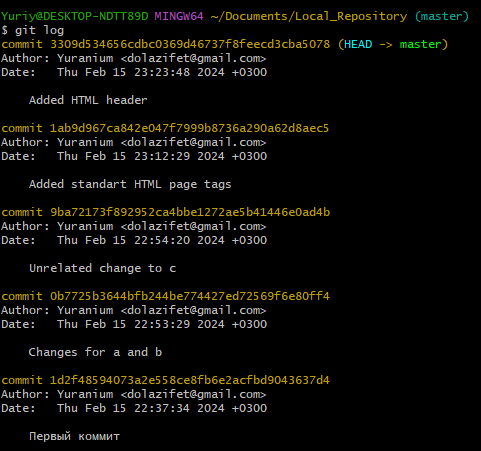


Рисунок 9 – Использование команды git log

Применения различных опций к команде git log (рисунок 10).

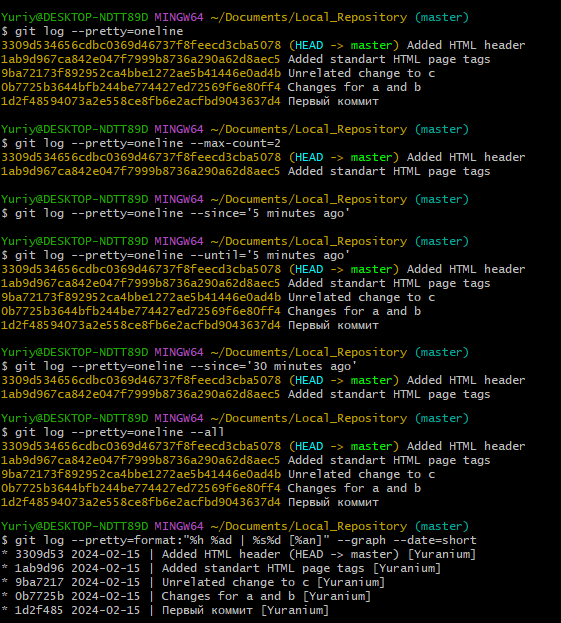


Рисунок 10 – Использование команды git log с опциями

Просмотр предыдущих версий файлов (рисунок 11).



Рисунок 11 – Использование команды git checkout

Создание тэгов при помощи команды git tag (рисунок 12).

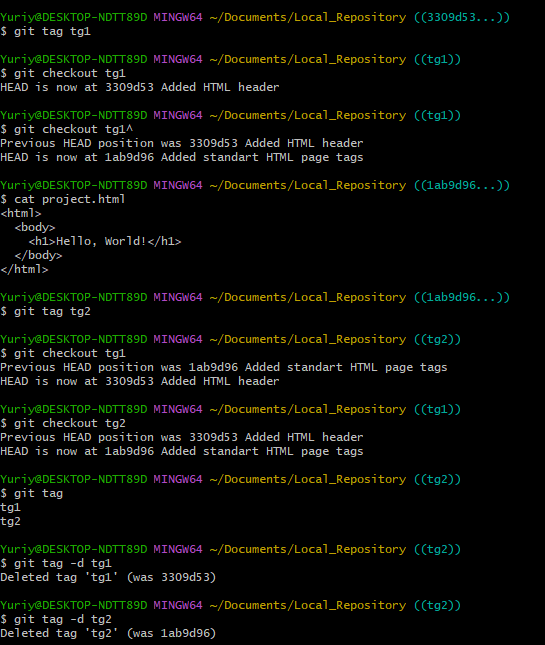


Рисунок 12 – Работа с тэгами

Отмена изменений в git репозитории перед созданием коммита (рисунок 13).

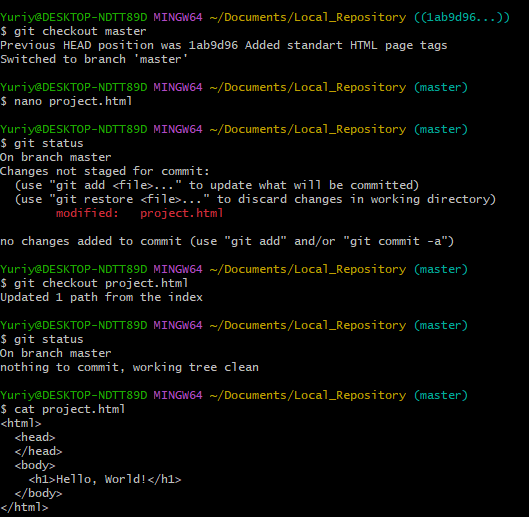


Рисунок 13 – Отмена изменений перед их созданием коммита

Отмена изменений после создания коммита (рисунок 14).

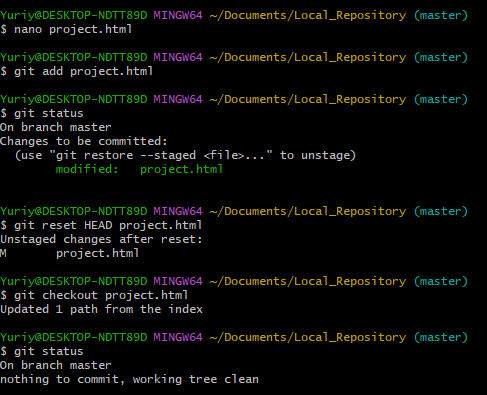


Рисунок 14 – Отмена изменений после создания коммита

Отмена коммита (рисунок 15).

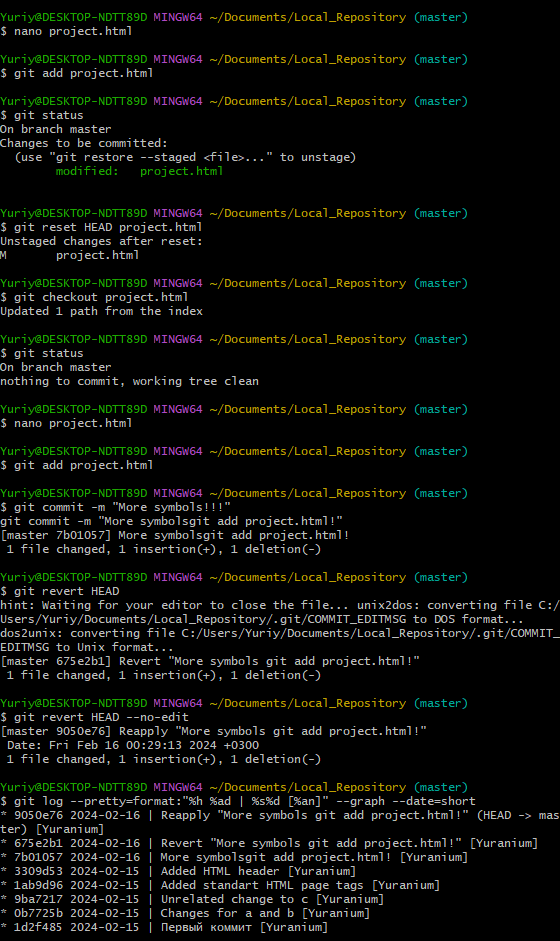


Рисунок 15 – Отмена коммита

Часть 2.

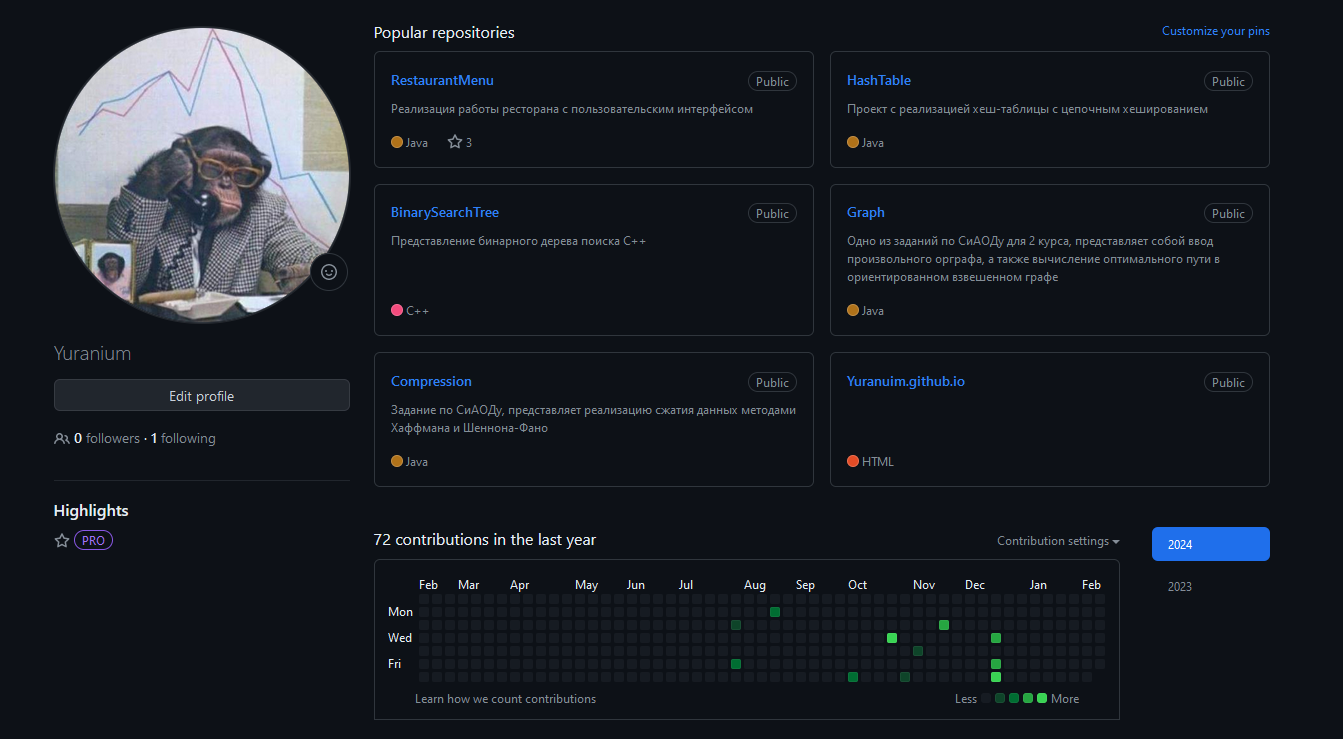


Рисунок 16 – Аккаунт на GitHub

Создание репозитория на GitHub, рис. 17, 18

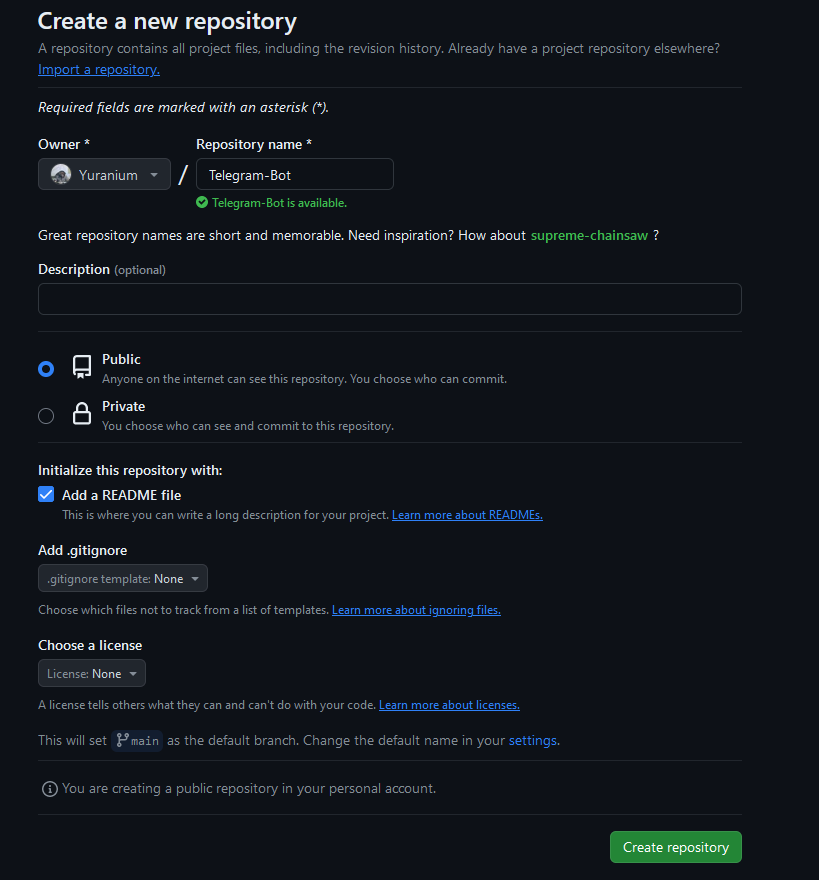
**

Рисунок 17 - Создание репозитория на GitHub (1)

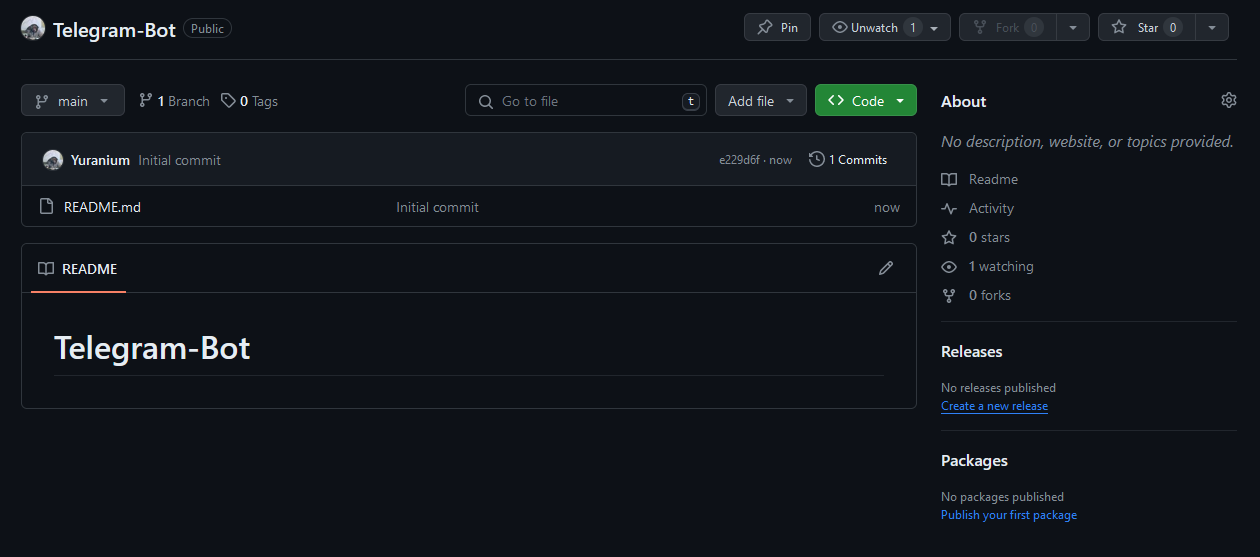
**

Рисунок 18 - Создание репозитория на GitHub (2)

Создание репозитория на локальной машине, рис. 19

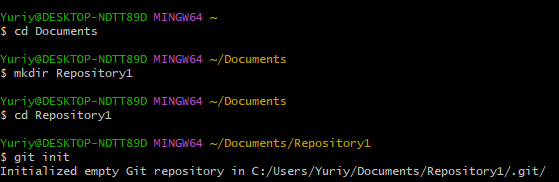
**

Рисунок 19 – Создание репозитория на локальной машине.

Создание нескольких файлов на локальной машине при помощи консоли,

рис. 20

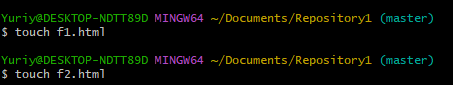
**

Рисунок 20 - Создание нескольких файлов на локальной машине при помощи консоли

Создание SSH-ключа для авторизации

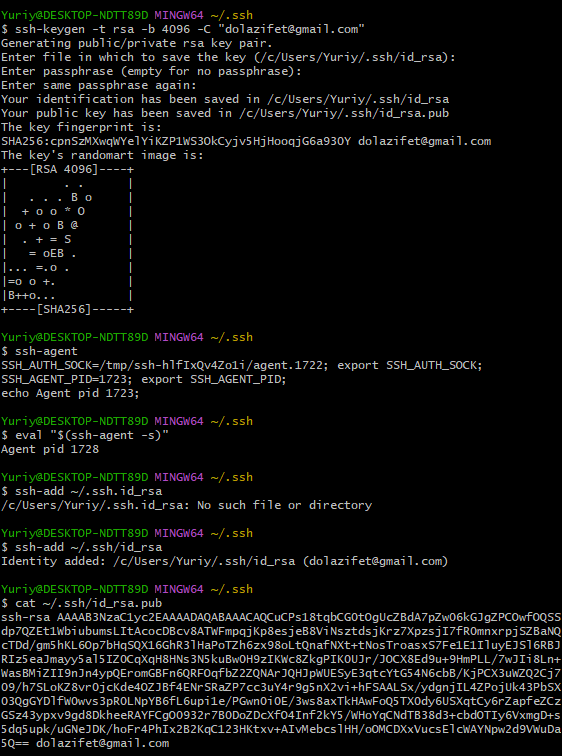
**

Рисунок 21 – Создание SSH ключа

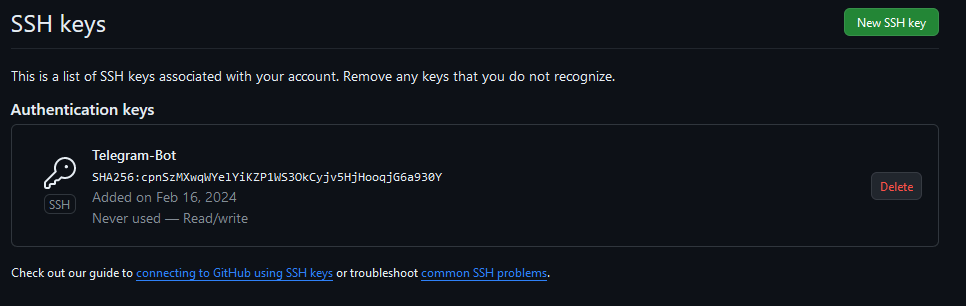
**

Рисунок 22 – Привязка SSH ключа к GitHub

Связь репозитория на локальной машины с репозиторием на GitHub при помощи консоли

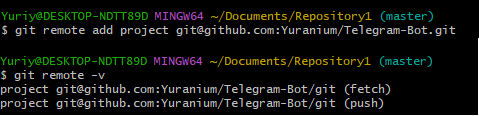
**

Рисунок 23 – Связь локального репозитория с репозиторием на GitHub

Создайте новую ветку в репозитории с помощью команды, произведите в ней какие-нибудь изменения, а после слейте с веткой master

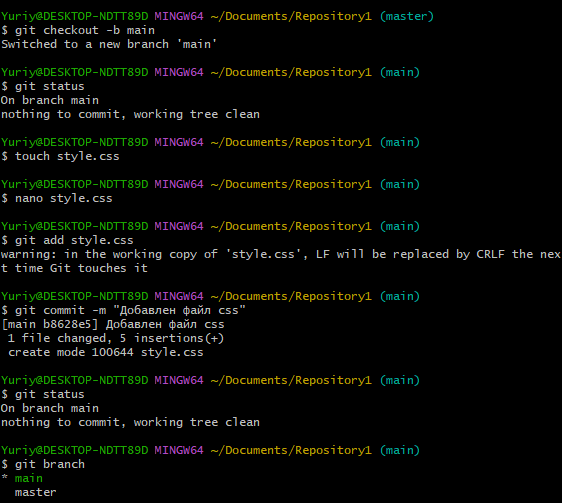
**

Рисунок 24 – Создание новой ветки и добавление туда нового файла

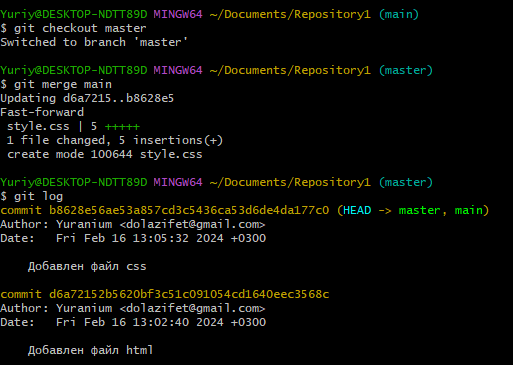
**

Рисунок 25 – Слияние ветки main в master

Клонирование непустого удаленного репозитория на локальную машину

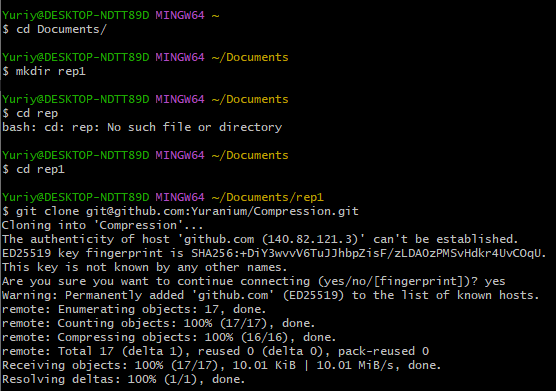
**

Рисунок 26 – Клонирование непустого удаленного репозитория на локальную машину

Создание новой ветки и выведение списка всех веток

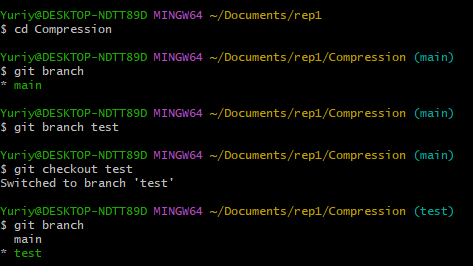
**

Рисунок 27 – Создание новой ветки и выведение списка всех веток

Произведение 3 коммитов в новой ветке в разных файлы

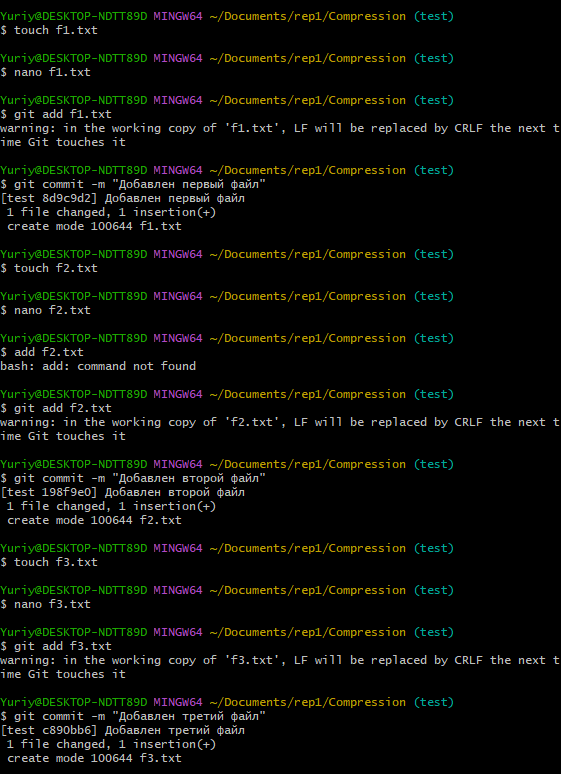
**

Рисунок 28 – Три коммита в ветке test

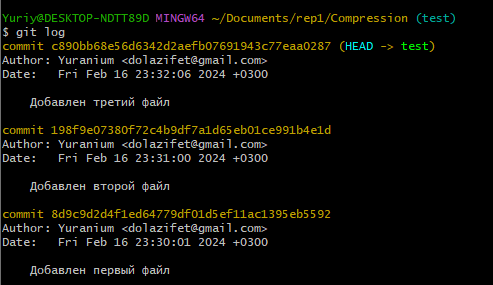
**

Рисунок 29 – История коммитов в ветке test

Выгрузка изменения в удаленный репозиторий

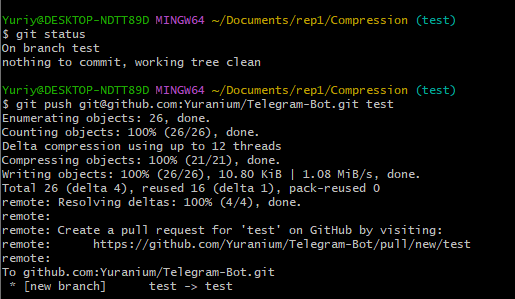
**

Рисунок 30 – Выгрузка репозитория в удалённый репозиторий

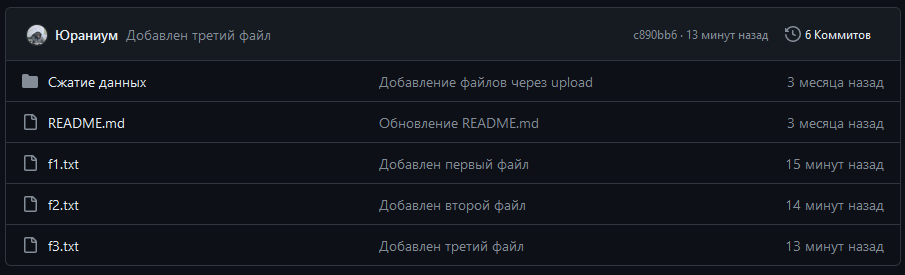
**

Рисунок 31 – Результат выгрузки репозитория на удаленный сервер.

Произведите revert предпоследнего коммита в новой ветке

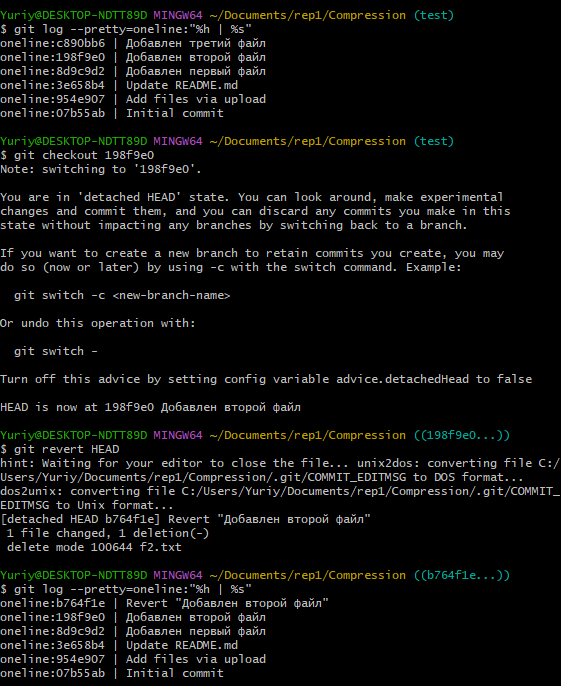
**

Рисунок 32 – Отмена предпоследнего коммита

Вывод в консоли различий между веткой master и новой веткой

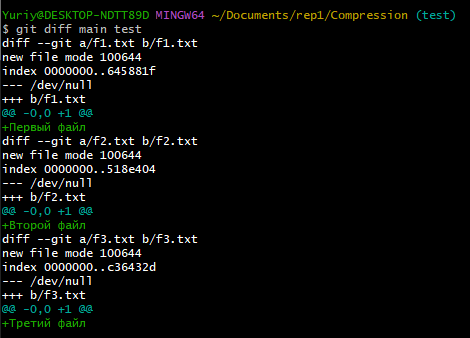
**

Рисунок 33 – Различия между двумя ветками.

Слияние новой ветки с master при помощи merge

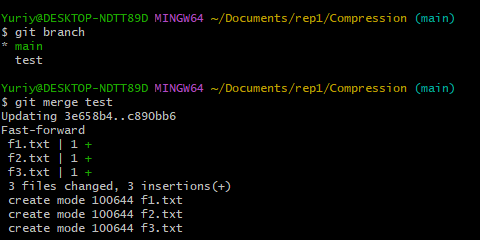
**

Рисунок 34 – Слияние ветки test с main

Часть 3.

Форк репозитория в соответствии с моим вариантом

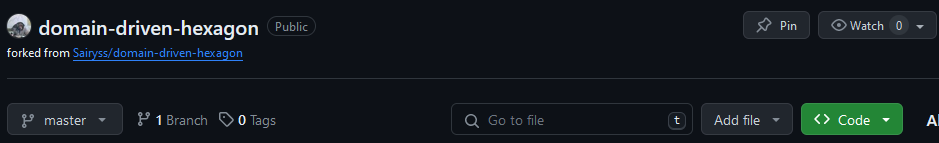
**

Рисунок 35 – Форк репозитория по ссылке

Клонирование данного репозитория на локальную машину

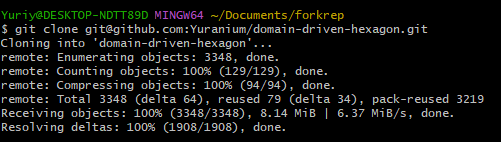
**

Рисунок 36 – Клонирование данного репозитория

Создание двух веткок branch1 и branch2 от последнего коммита в master'е

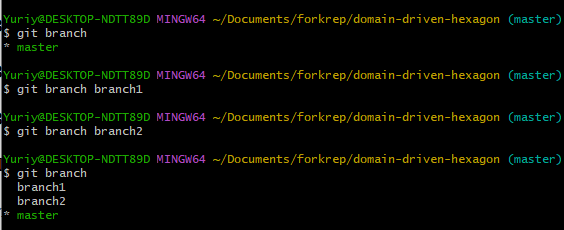
**

Рисунок 37 – Создание двух веток от ветки master

Проведение 3ёх коммитов в каждую из веток, которые меняют один и тот же кусочек файла

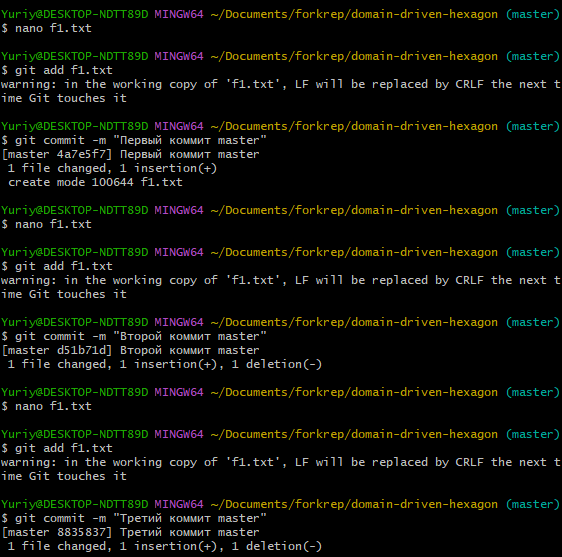
**

Рисунок 38 – Три коммита в ветке master

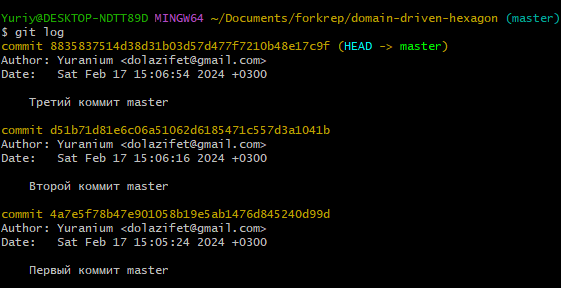
**

Рисунок 39 – История данных коммитов в ветке master

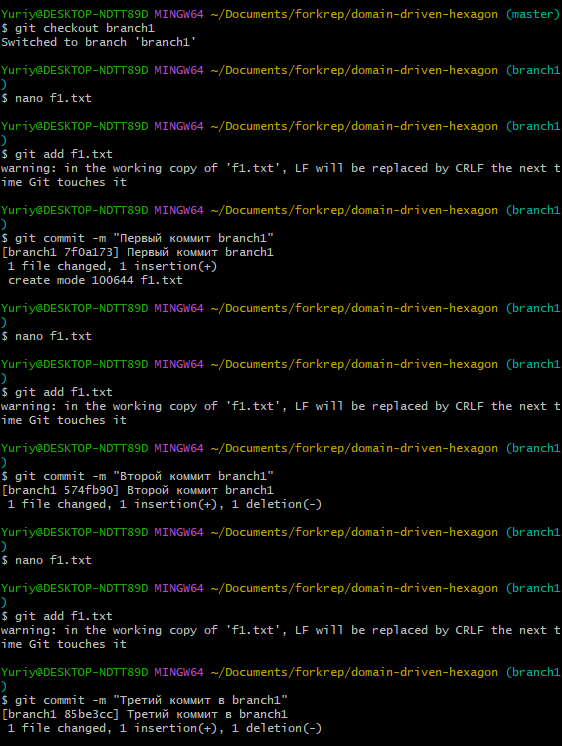
**

Рисунок 40 – Три коммита в ветке branch1

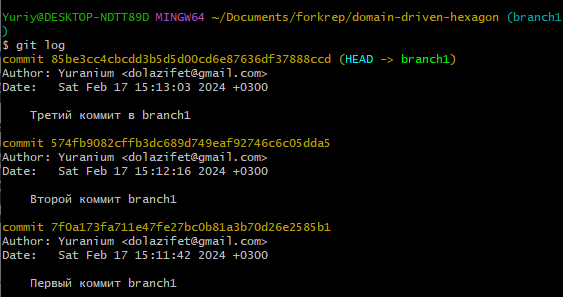
**

Рисунок 41 – История данных коммитов в ветке branch1

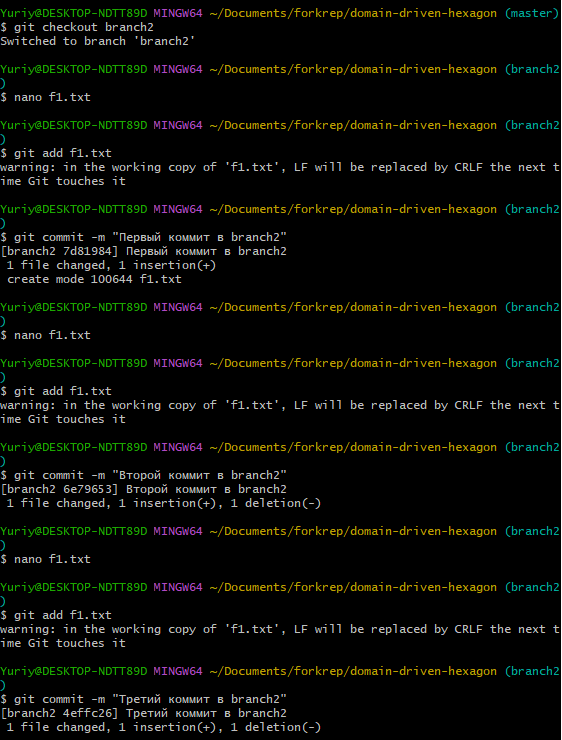
**

Рисунок 42 – Три коммита в ветке branch2

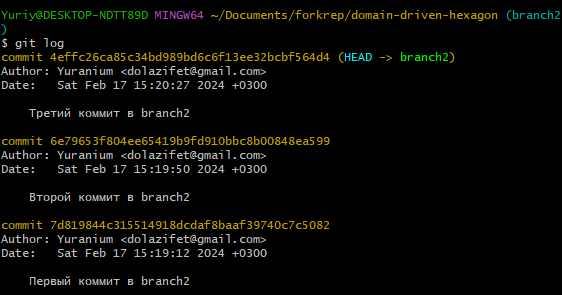
**

Рисунок 43 – История данных коммитов ветке branch2

Выполнение слияния ветки branch1 в ветку branch2, разрешив конфликты при этом

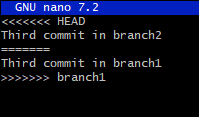
**

Рисунок 44 – Конфликт при слиянии веток в файле

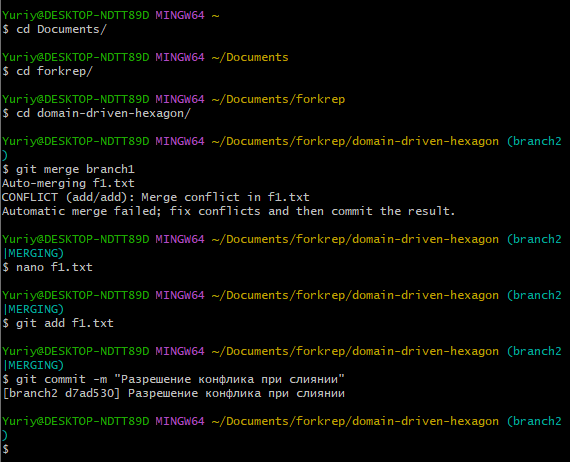
**

Рисунок 45 – Слияние ветки branch1 в branch2 с разрешением конфликта

После разрешения конфликта процесс слияния продолжился автоматически.

Выгрузка всех изменений во всех ветках в удаленный репозиторий

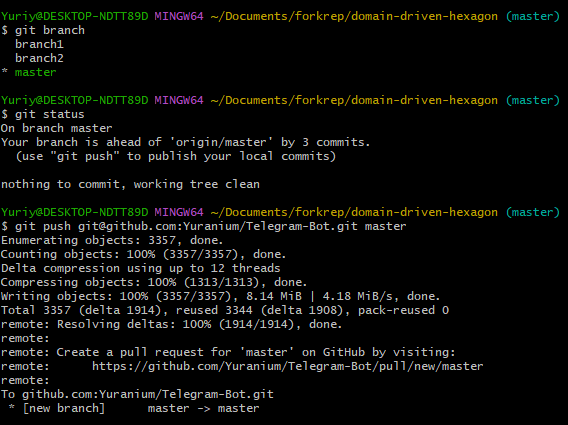
**

Рисунок 46 – Выгрузка изменений в удаленный репозиторий из ветки master.

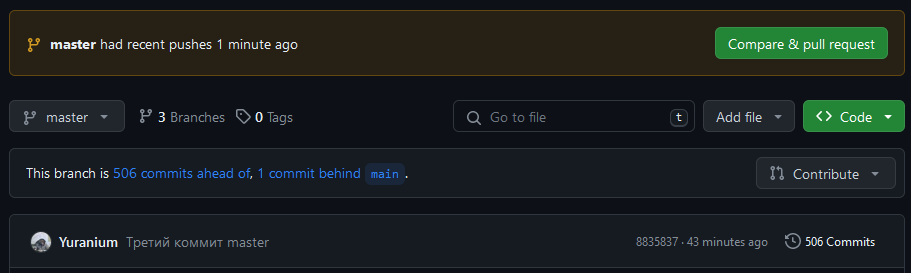
**

Рисунок 47 – Добавление на GitHub.

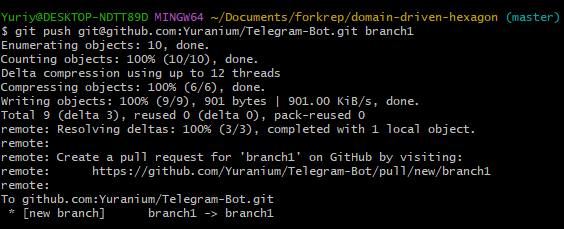
**

Рисунок 48 – Выгрузка изменений в удаленный репозиторий из ветки branch1.

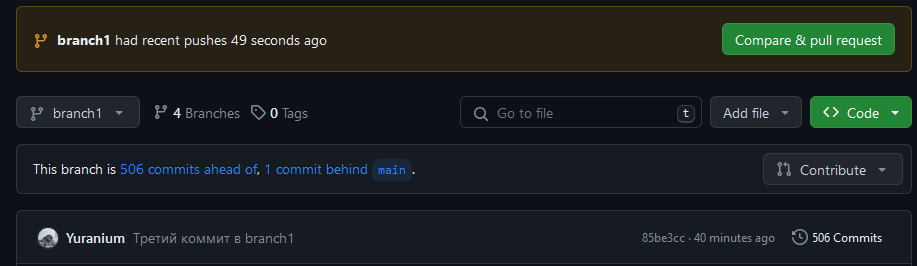
**

Рисунок 49 – Добавление на GitHub.

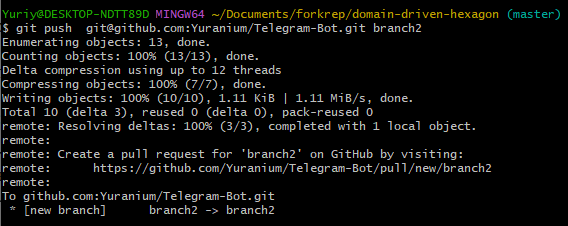
**

Рисунок 50 – Выгрузка изменений на удаленный репозиторий из ветки branch2.

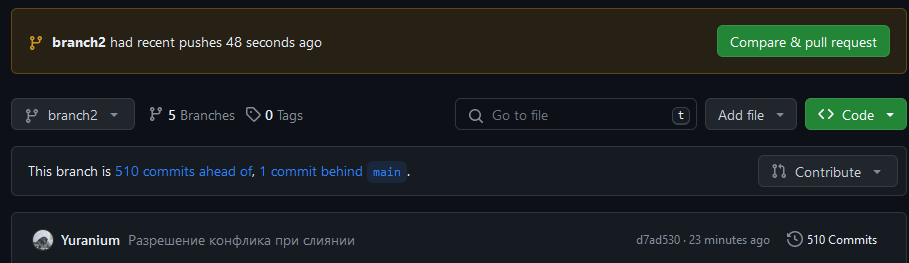
**

Рисунок 51 – Добавление на GitHub.

Проведение еще 3ёх коммитов в ветку branch1

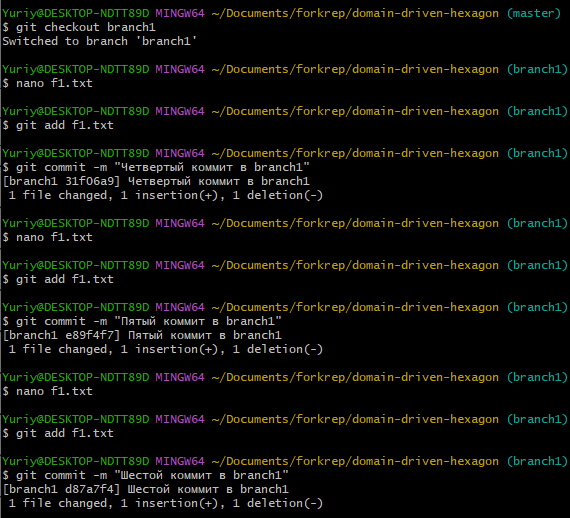
**

Рисунок 52 – Три коммита в ветке branch1.

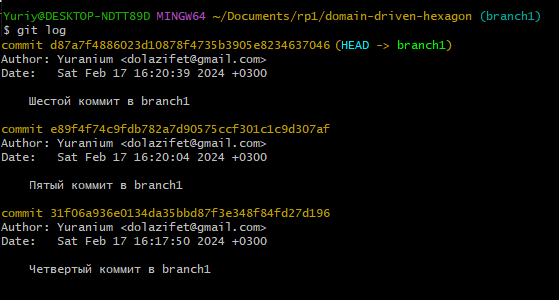
**

Рисунок 53 – История данных коммитов в ветке branch1*.*

Клонирование репозитория еще раз в другую директорию

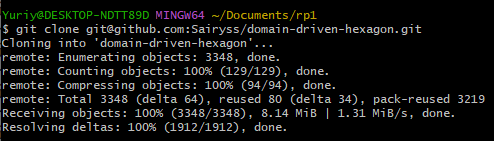
**

Рисунок 54 – Повторное клонирование репозитория в другую директорию.

В новом клоне репозитории добавление еще 3ёх коммитов в ветку branch1

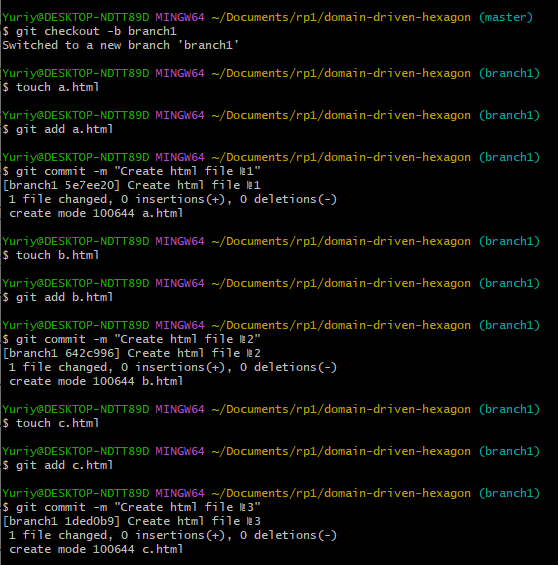
**

Рисунок 55 – Создание трех коммитов в ветке branch1 в новой копии репозитория.

Выгрузка всех изменений из нового репозитория в удаленный репозиторий

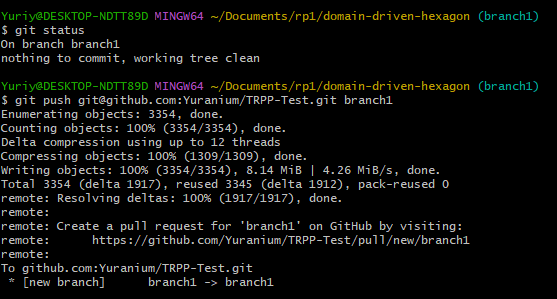
**

Рисунок 56 – Выгрузка изменений в клоне репозитория в удаленный репозиторий из ветки branch1.

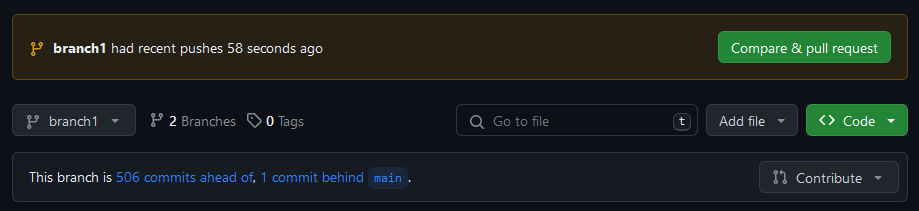
**

Рисунок 57 – Добавление на GitHub.

Возврат в старый клон с репозиторием и выгрузка изменения с опцией –force

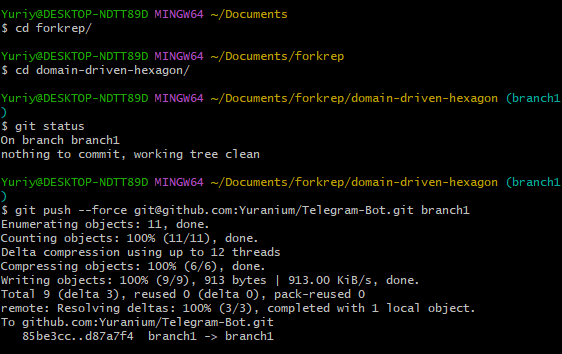
**

Рисунок 57 – Выгрузка локального репозитория в удалённый с опцией –force, которая делает принудительный выгруз репозитория.

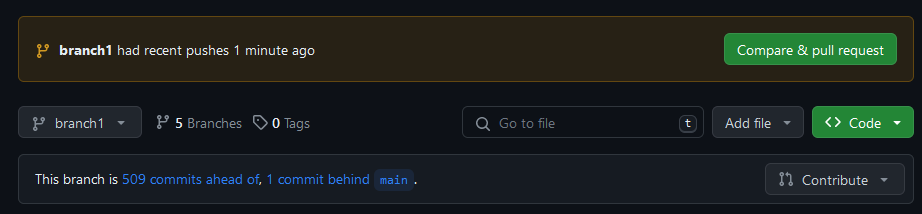
**

Рисунок 58 – Результат принудительной загрузки на GitHub.

Получение всех изменений в новом репозитории

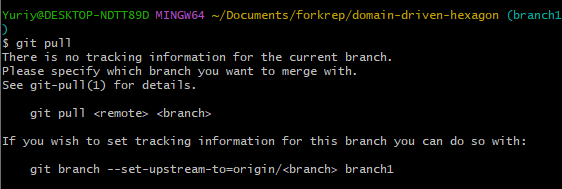
**

Рисунок 59 – Результат команды git pull.

Контрольные вопросы.

1. Что делает команда git status? Данная команда отображает список измененных, добавленных и удаленных файлов.
2. Что делает команда git add? Команда git add добавляет изменение в рабочий каталог в промежуточную область. Она сообщает Git, что в проекте есть несколько обновлений, которые пользователь хочет зафиксировать.
3. Что делает команда git log? По умолчанию (без аргументов) git log перечисляет коммиты, сделанные в репозитории в обратном к хронологическому порядке — последние коммиты находятся вверху.
4. Что делает команда git diff? Для вывода изменений в файлах по сравнению с последним коммитом, используется git diff без параметров.
5. Что сделает команда "git branch" без какого-либо параметра? При запуске без параметров, вы получите простой список имеющихся у вас веток.
6. Как сделать ветку с названием my\_branch? Необходимо ввести команду git branch my\_branch.
7. Как сделать коммит для ветки my\_branch? Необходимо перейти в ветку my\_branch при помощи команды git checkout my\_branch, далее произвести какие-то изменения, зафиксировать их, а после совершить коммит.
8. Что такое форк репозитория? Форк репозитория - это полноценная копия репозитория.

# Выводы

В ходе выполнение данной работы я научился использовать систему контроля версий – Git и GitHub, научился работать с Git в терминале для удобного размещения моих проектов.