|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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** | |
| **«Технологии разработки программных приложений»**  **Тема: «Системы контроля версий»** | |
|  | |
| Выполнил студент группы ИКБО-30-22 | Сенькевич Г.Д. |
| Принял преподаватель | Петрова А.А. |

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

Москва 2024

**СОДЕРЖАНИЕ**

[Задание 3](#_Toc160181504)

[Выполнение заданий 5](#_Toc160181505)

[Часть 1. Основные команды Git 5](#_Toc160181506)

[Часть 2. Системы управления репозиториями 16](#_Toc160181507)

[Часть 3. Ветвление и оформление кода 27](#_Toc160181508)

[Ответы на вопросы 35](#_Toc160181509)

[Вывод 36](#_Toc160181510)

Задание

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

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.

|  |  |
| --- | --- |
| 9 | 1) Клонируйте непустой удаленный репозиторий на локальную машину  2) Создайте новую ветку и выведите список всех веток  3) Произведите 5 коммитов в новой ветке в разные файлы  4) Выгрузите изменения в удаленный репозиторий  5) Произведите revert 2-го и 4-го 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 (вариант 5)
  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

1. Получите все изменения в новом репозитории

Выполнение заданий

Часть 1. Основные команды Git

Были прочитаны все материалы на GitHowTo (рис. 1).

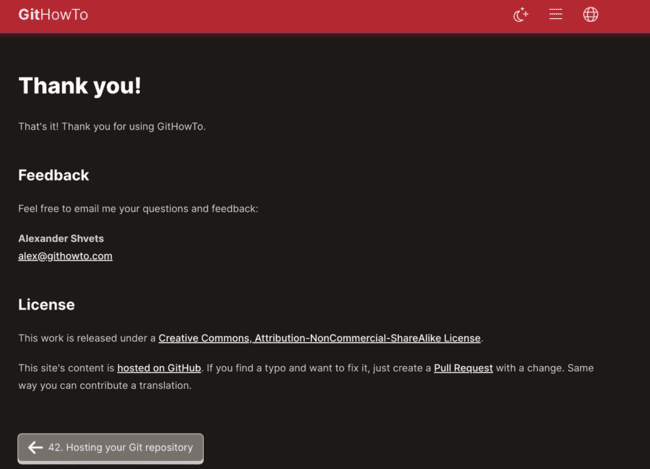


Рисунок 1 – Прохождение GitHowTo

Произведём настройку Git (уже установлен).

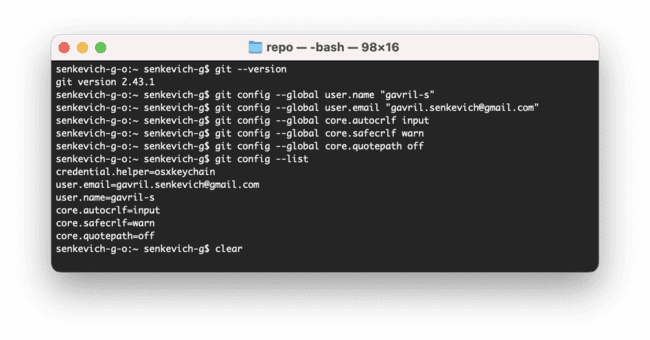


Рисунок 2 – Настройка Git

Создадим директорию, в которой будет находится репозиторий, создадим в ней файл index.html.

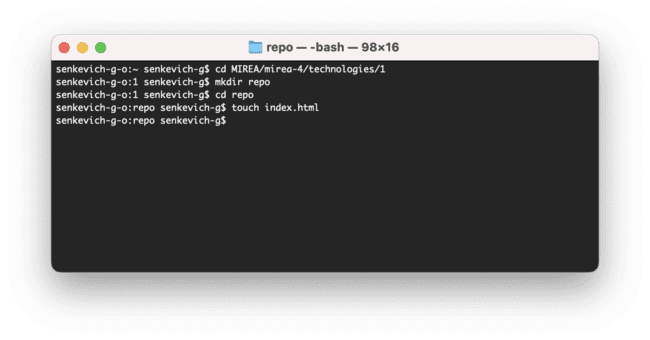


Рисунок 3 – Созданине директории и файла

Заполним файл index.html, используя утилиту nano.

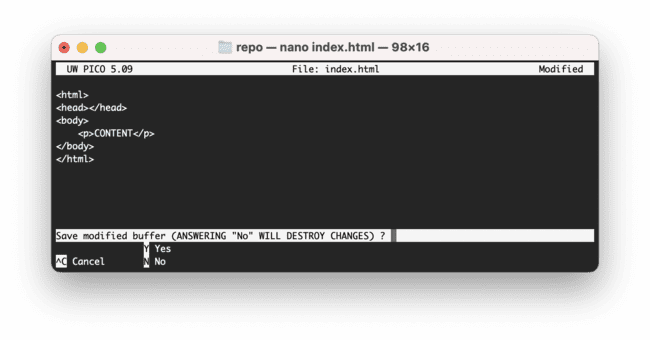


Рисунок 4 – Заполнение файла

Инициализируем репозиторий.

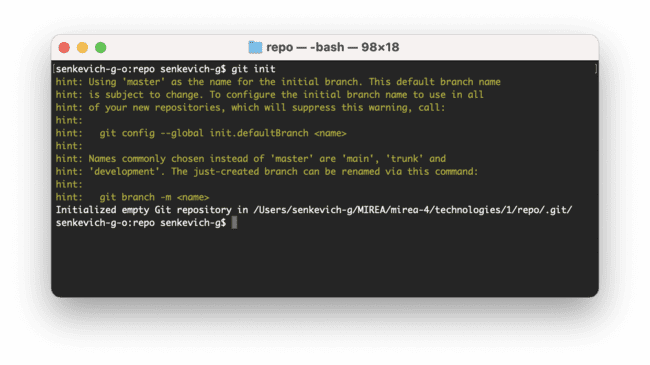


Рисунок 5 – Инициализация репозитория

Проиндексируем созданный ранее файл, создадим коммит и проверим статус репозитория.

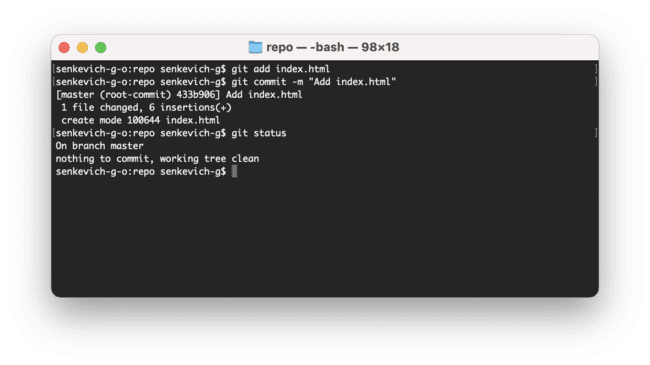
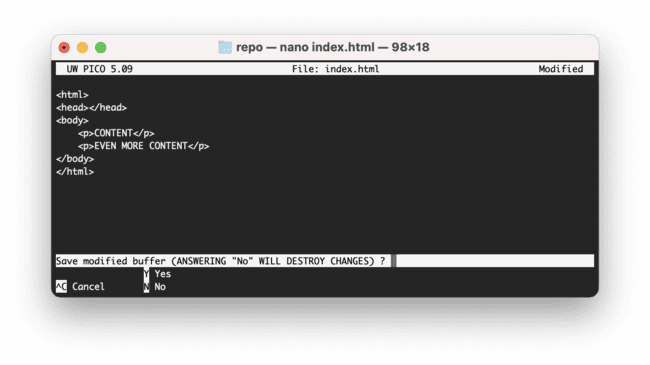
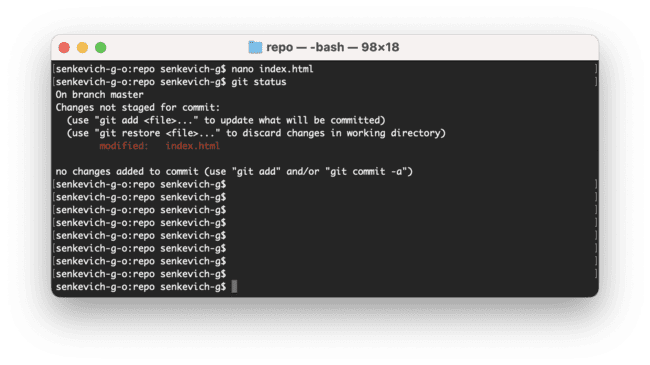


Рисунок 6 – Первый коммит

Внесём изменения в файл с помощью nano, проверим статус репозитория и проиндексируем изменения.

 Рисунок 7 – Изменение файла

Рисунок 8 – Статус репозитория

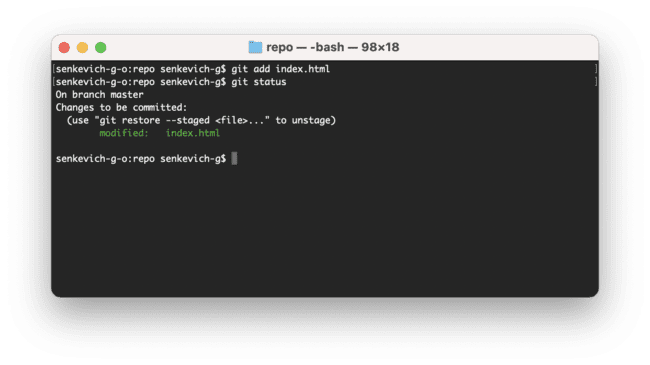
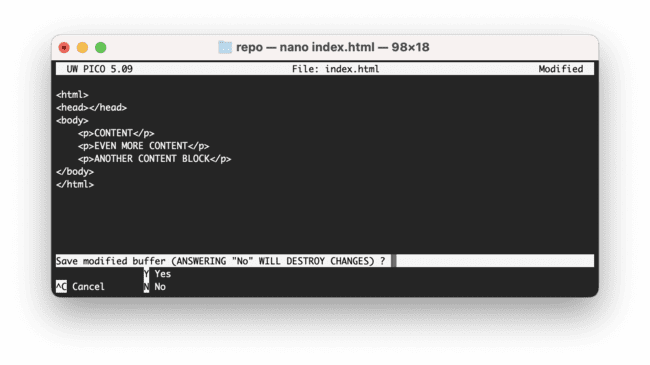


Рисунок 9 – Индексация изменений

Проведём ещё одно изменение, проверим статус репозитория, сделаем второй коммит и снова проверим статус, после чего проиндексируем изменения и сделаем третий коммит.

 Рисунок 10 – Изменение файла

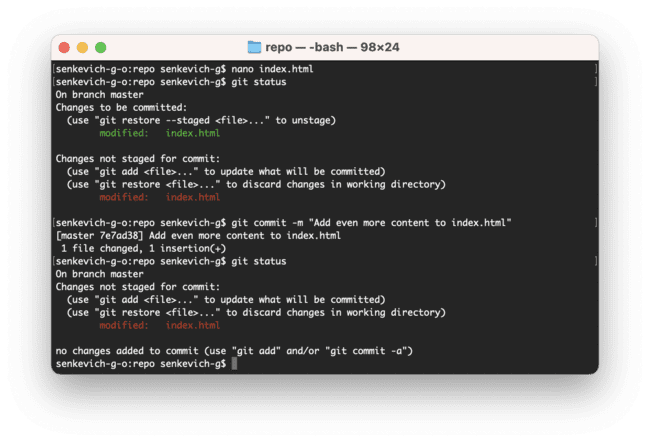
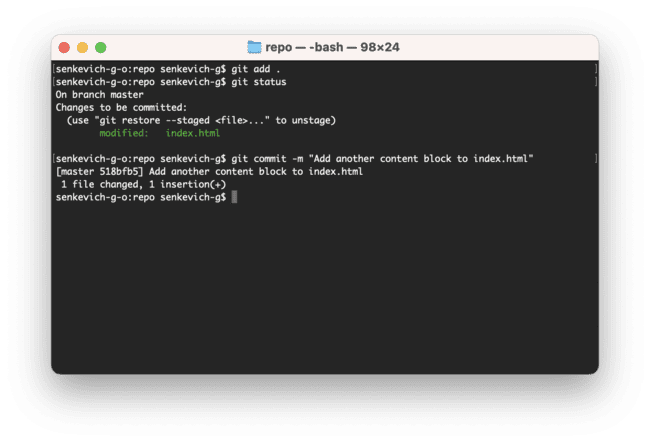


Рисунок 11 – Второй коммит

Рисунок 12 – Третий коммит

Посмотрим историю коммитов с помощью команды git log. Ознакомимся с некоторыми параметрами команды.

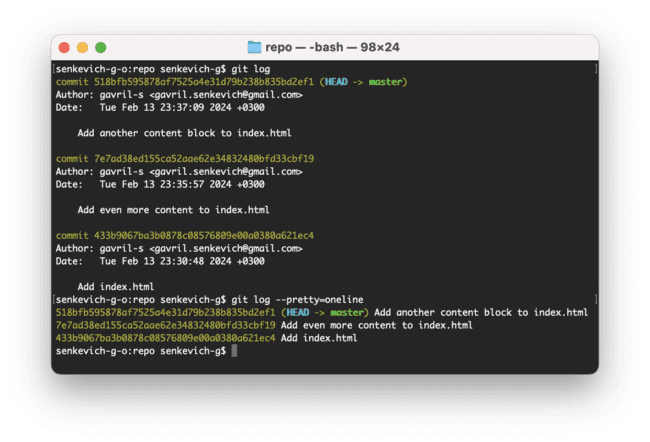
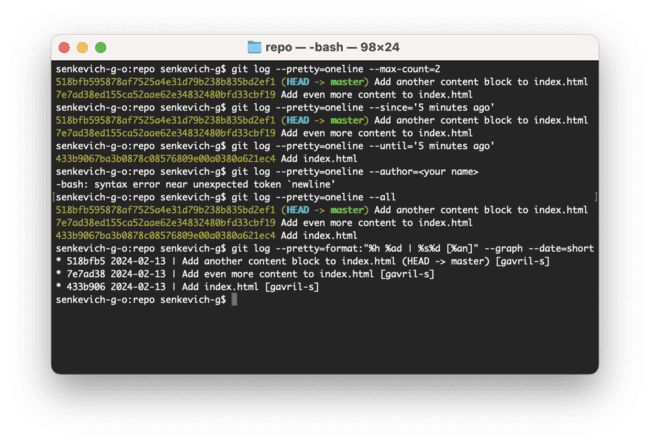


Рисунок 13 – История коммитов Рисунок 14 – Параметры команды git log

Вернём каталог к одному из предыдущих состояний, выполним проверку изменения состояния и вернёмся обратно на ветку master.

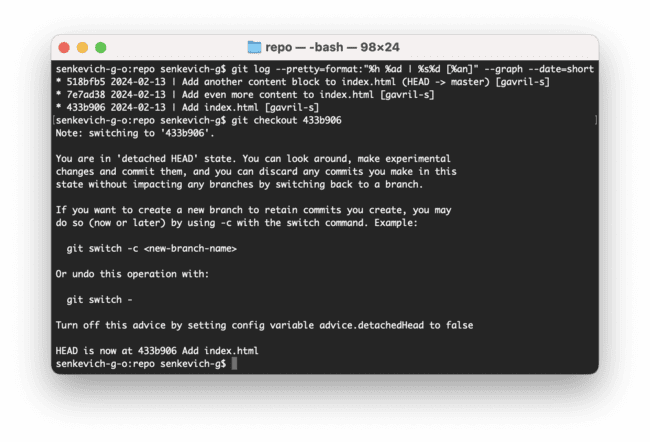


Рисунок 15 – Возврат в предыдущее состояние

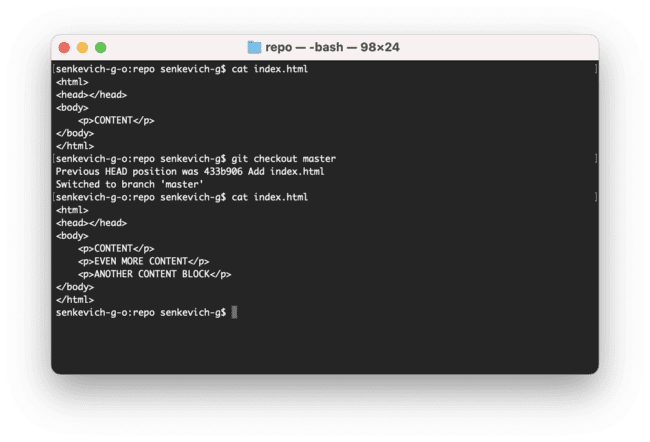


Рисунок 16 – Проверка состояния файла и возврат обратно

Создадим тэги v1 и v1-beta, переключимся с одного на другой.

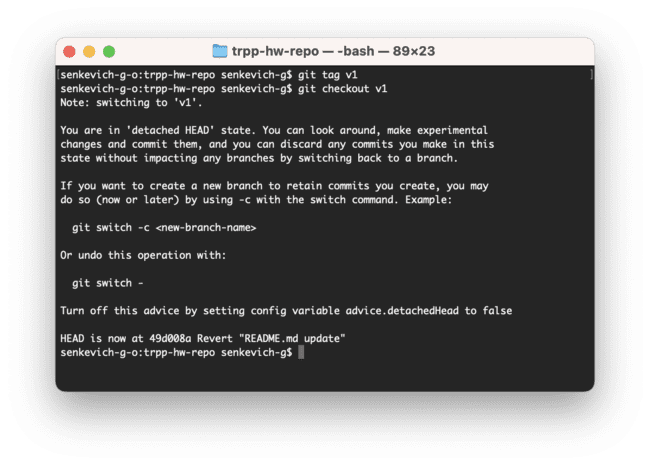


Рисунок 17 – Создание тэга v1

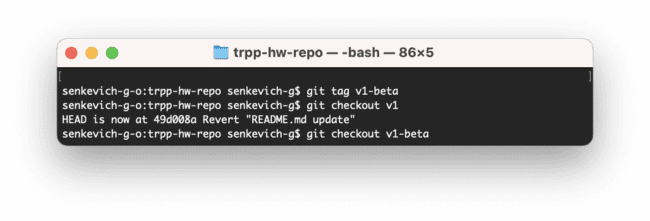


Рисунок 18 – Создание тэга v1-beta



Рисунок 19 – Список тэгов

Произведём ещё одно изменение, проверим статус репозитория, а потом отменим его с помощью команды git checkout.

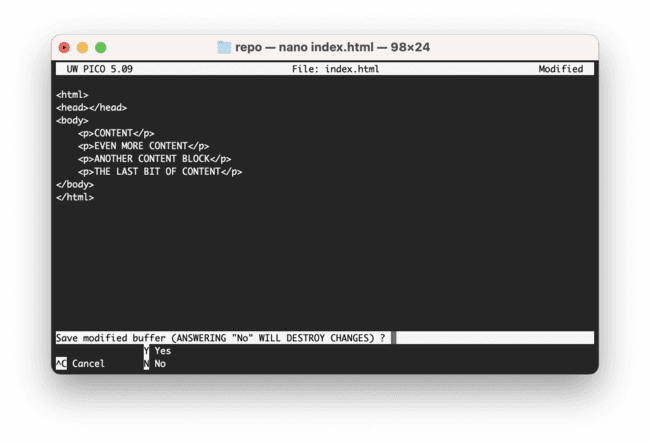


Рисунок 20 – Изменение файла

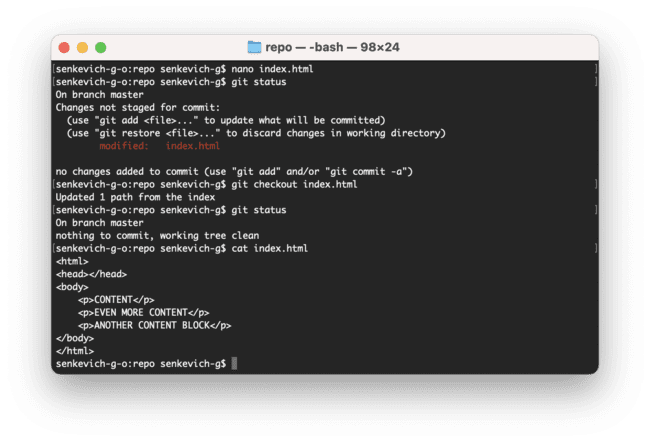


Рисунок 21 – Проверка состояния репозитория и отмена изменений

Произведём те же самые изменения в файле и отменим их, но на этот раз уже после индексирования.

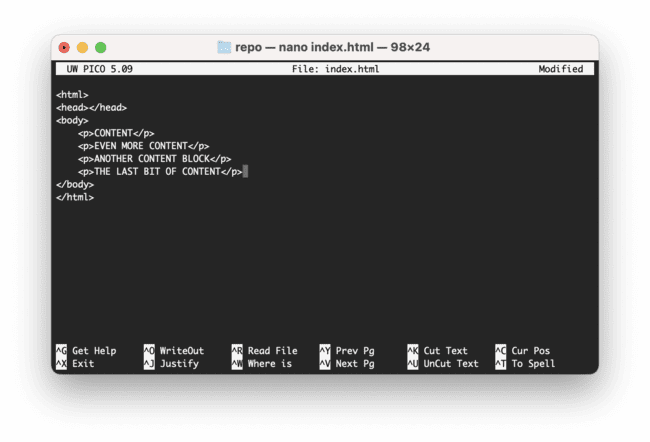
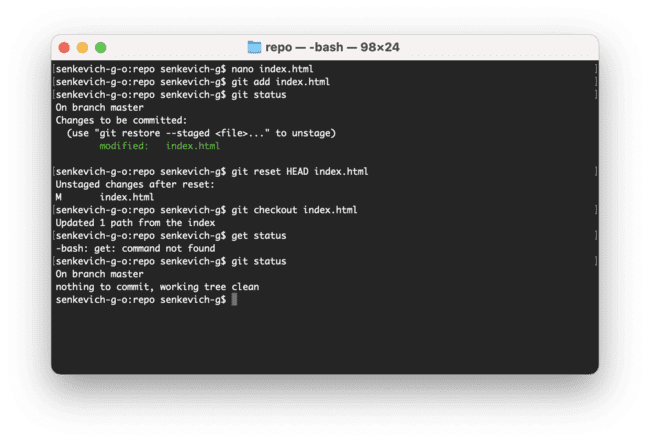


Рисунок 22 – Изменение файла

 Рисунок 23 – Проверка состояния репозитория и отмена изменений

Произведём ещё один коммит в репозиторий, а потом отменим его с помощью git revert.

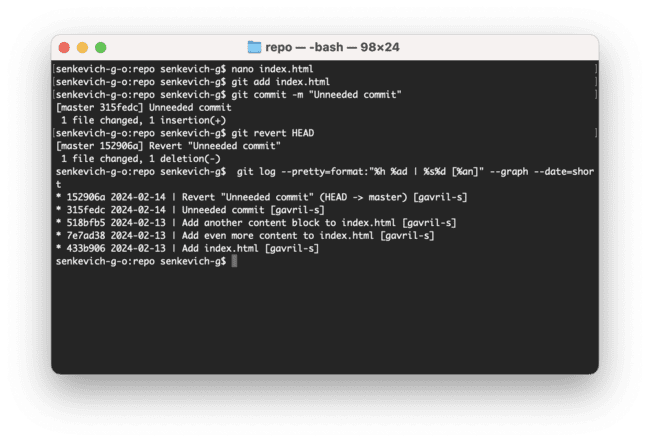


Рисунок 24 – Новый коммит и его отмена

Часть 2. Системы управления репозиториями

Откроем аккаунт на GitHub (уже создан).

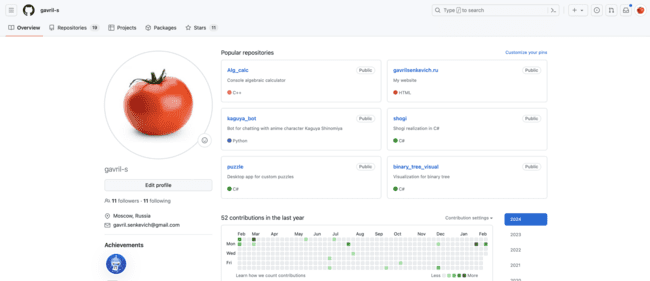


Рисунок 25 – Аккаунт на GitHub

Создадим новый ssh-ключ и добавим его к своему аккаунту на GitHub.

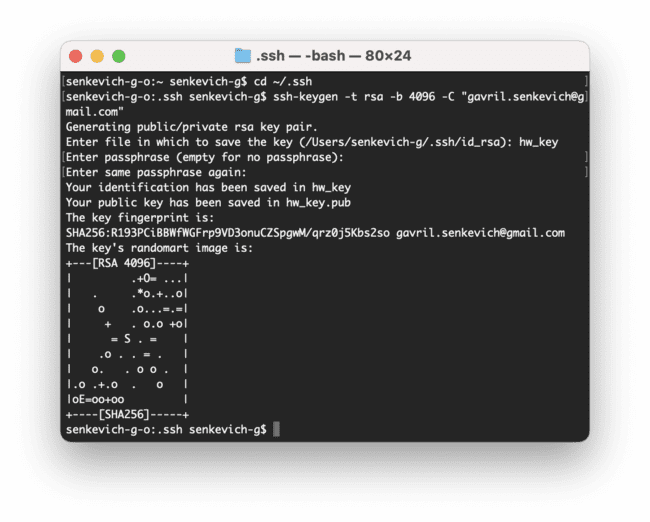


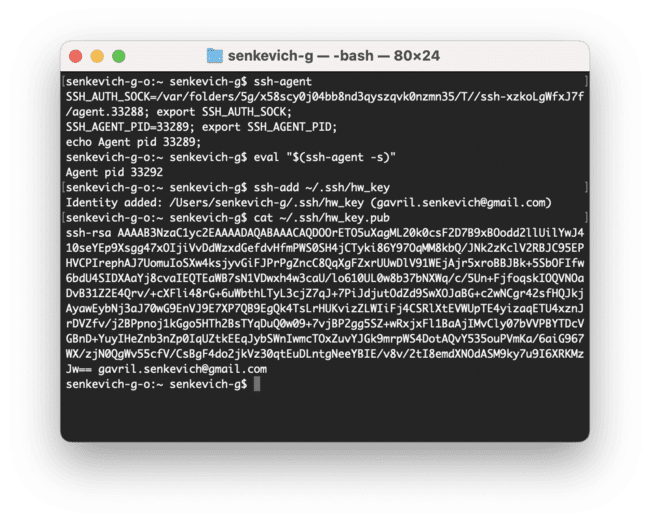
Рисунок 26 – Создание ssh-ключа

Рисунок 27 – Создание ssh-ключа

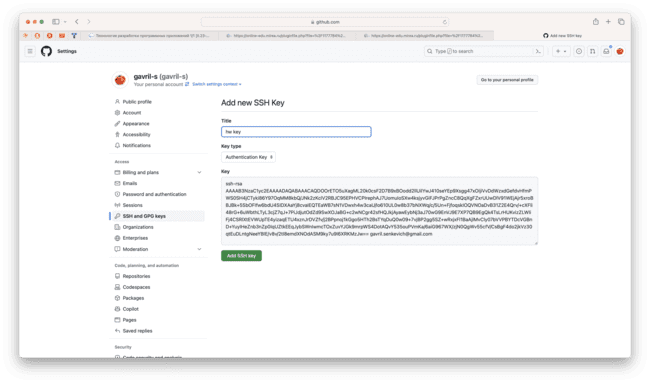


Рисунок 28 – Добавление ssh-ключа на GitHub

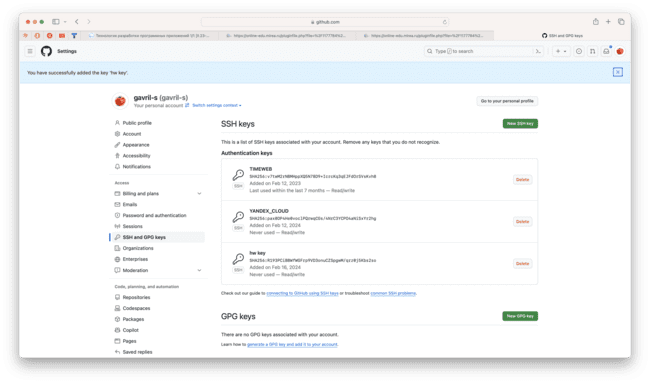


Рисунок 29 – Добавление ssh-ключа на GitHub

Создадим новый локальный репозиторий.

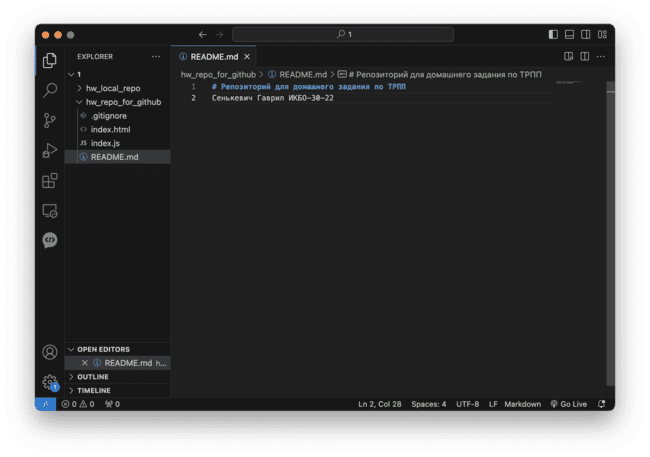


Рисунок 30 – Создание локального репозитория

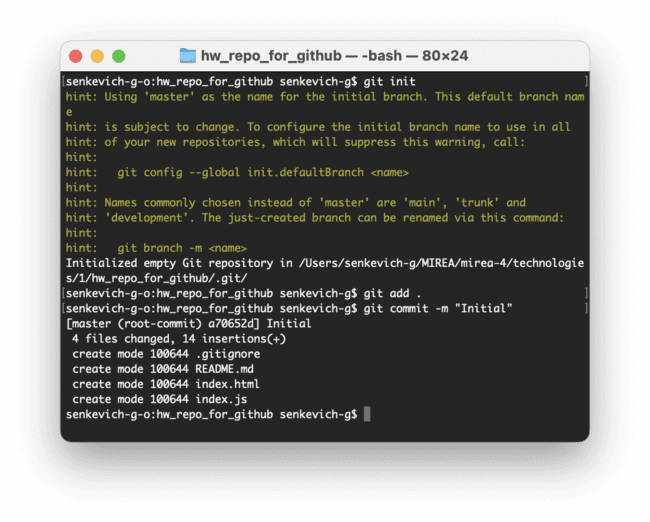


Рисунок 31 – Создание локального репозитория

Создадим удалённый репозиторий и свяжем его с локальным.

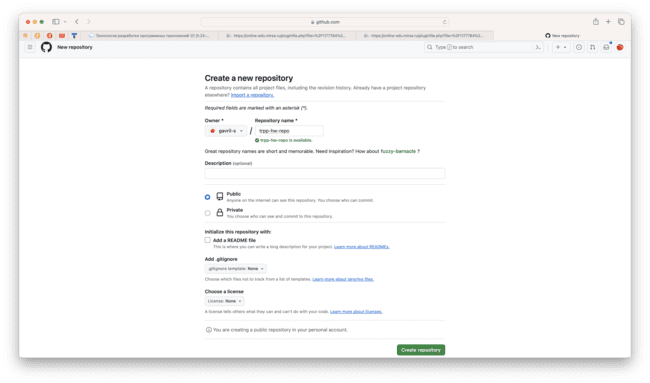


Рисунок 32 – Создание удалённого репозитория

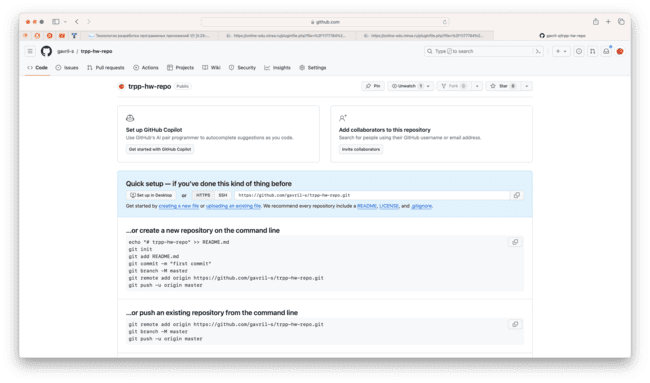


Рисунок 33 – Создание удалённого репозитория

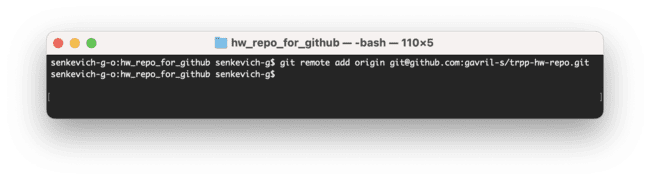


Рисунок 34 – Привязка локального репозитория к удалённому

Создадим в локальном репозитории новую ветку, внесём изменения, произведём коммит в новую ветку, после чего выполним merge с веткой master.

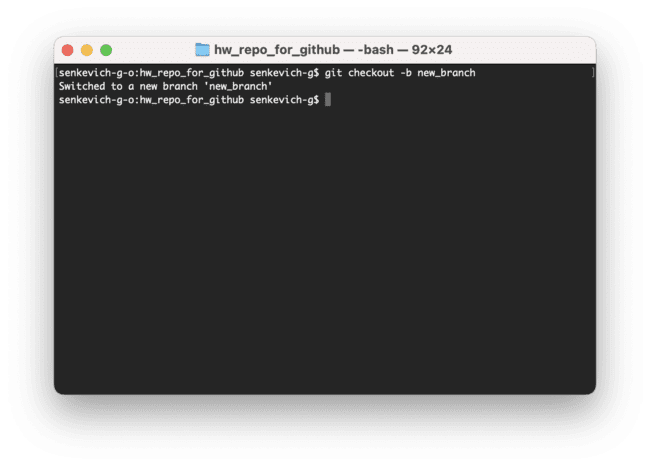
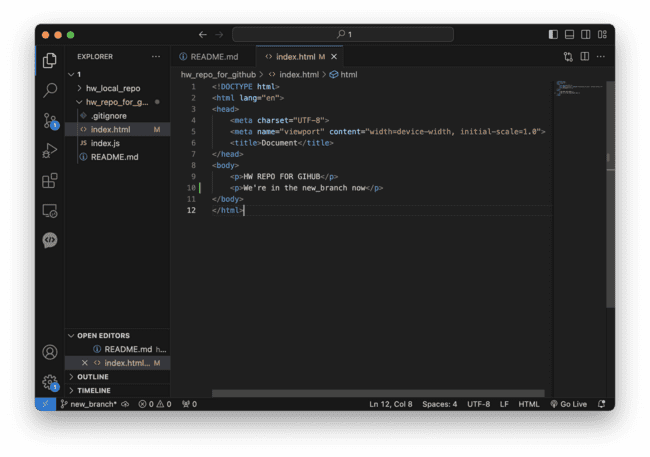
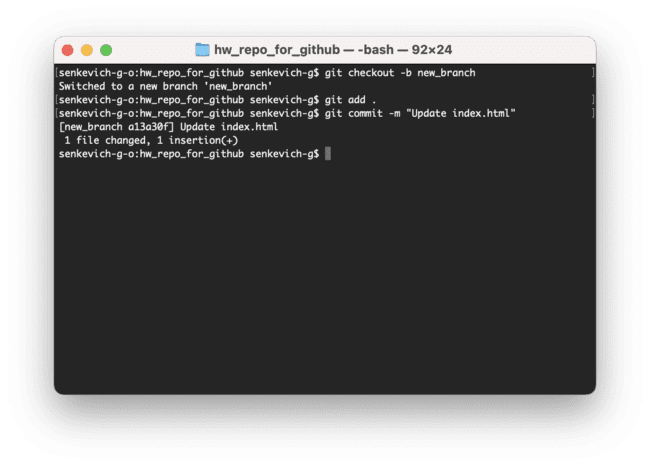
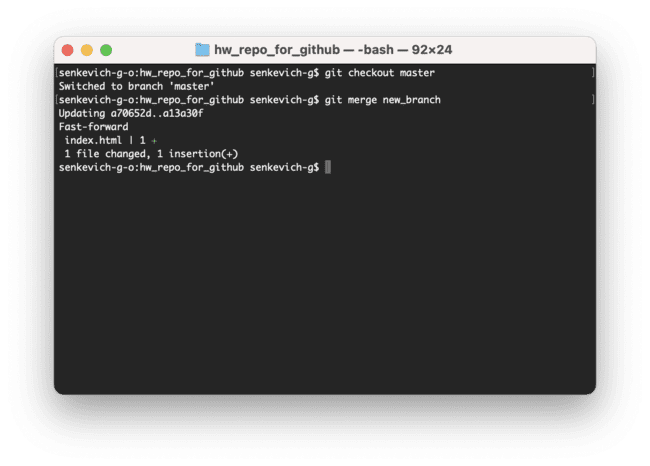


Рисунок 35 – Создание новой ветки

Рисунок 36 – Внесение изменений

Рисунок 37 – Фиксирование изменений (коммит)

Рисунок 38 – merge с веткой master

Выполним цепочку действий согласно варианту 9.

Склонируем удалённый репозиторий на локальную машину. Создадим новую ветку и выведем список всех веток.

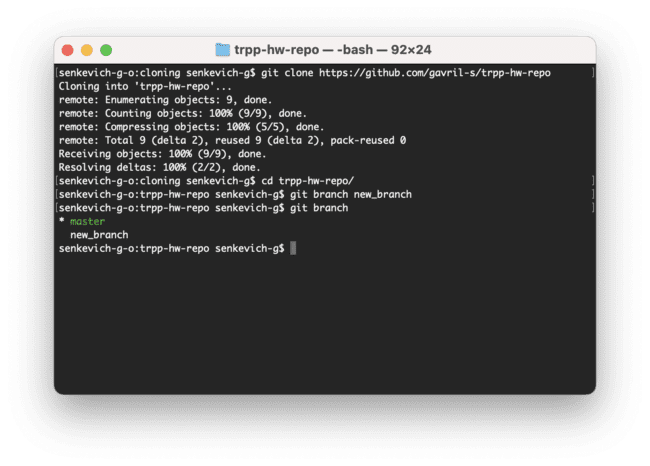


Рисунок 39 – Пункты 1-2 индивидуального варианта

Произведём 5 коммитов в новой ветке в разные файлы.

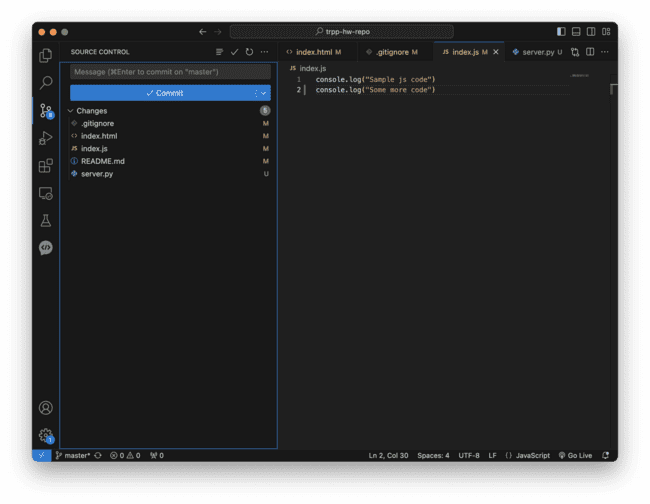


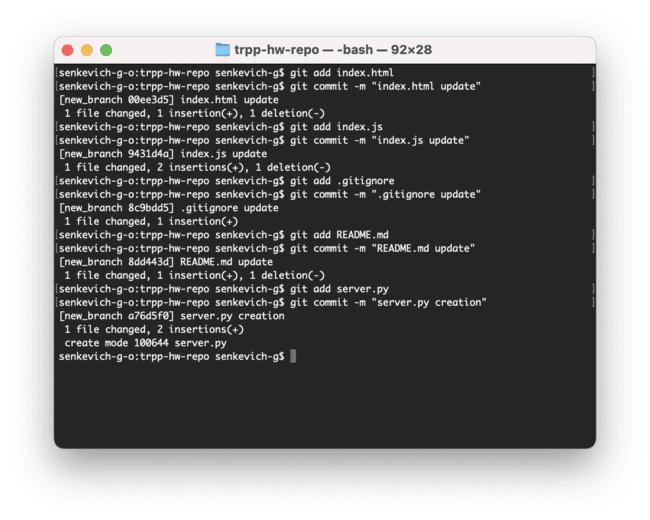
Рисунок 40 – Внесение изменений (пункт 3)

Рисунок 41 – Коммит изменений (пункт 3)

Выгрузим изменения в удалённый репозиторий.

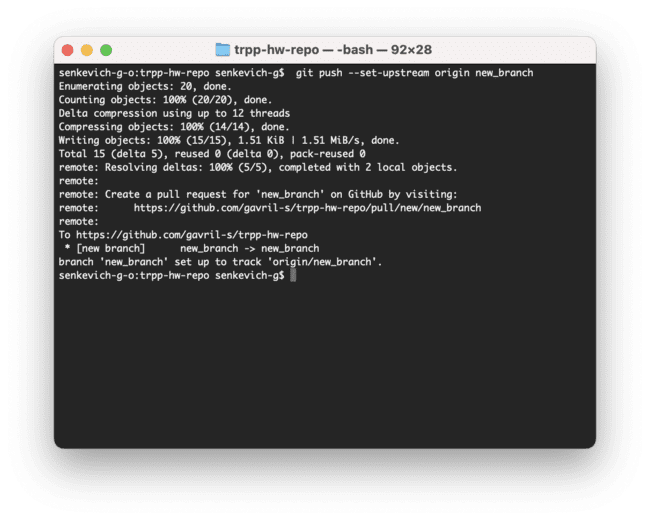


Рисунок 42 – Выгрузка изменений (пункт 4)

Произведём revert 2-го и 4-го коммита в новой ветке.

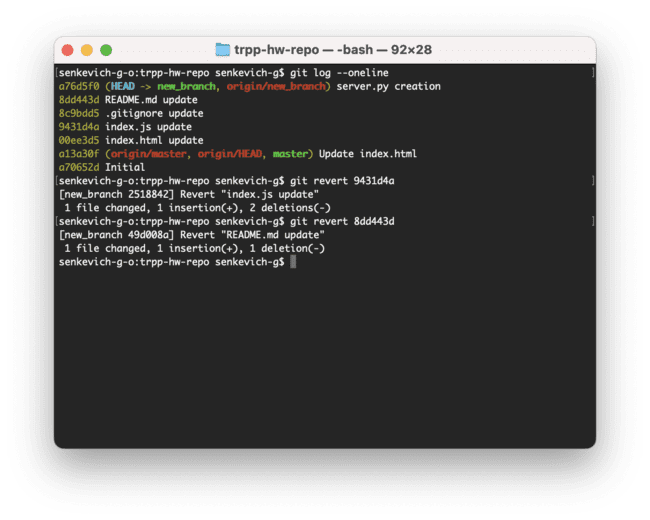


Рисунок 43 – revert (пункт 5)

Выведем различия между веткой master и новой веткой и произведём merge веток.

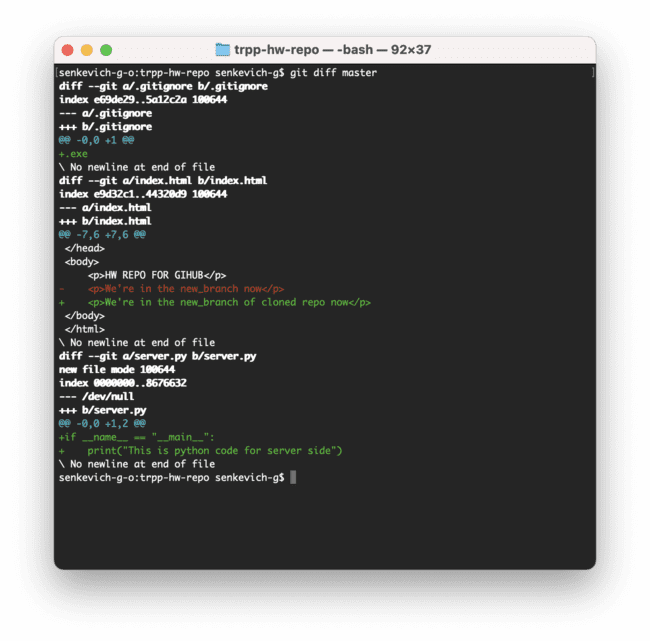


Рисунок 44 – Различия веток (пункт 6)

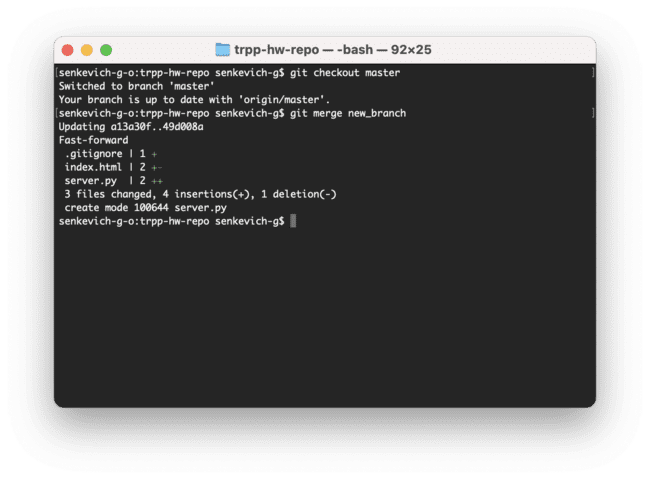


Рисунок 45 – merge веток (пункт 7)

Часть 3. Ветвление и оформление кода

Сделаем форк репозитория в соответствии с вариантом №7 и клонируем его на локальную машину.

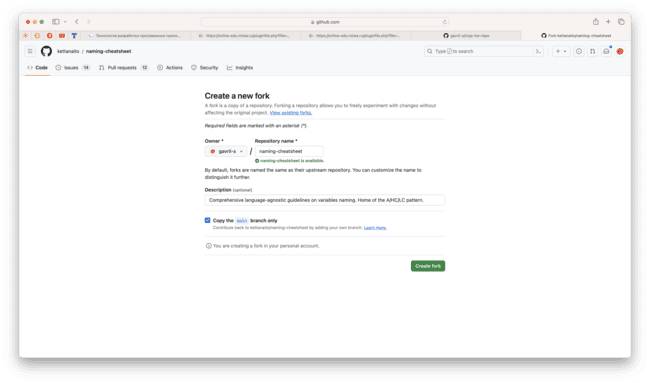


Рисунок 46 – Создание форка репозитория

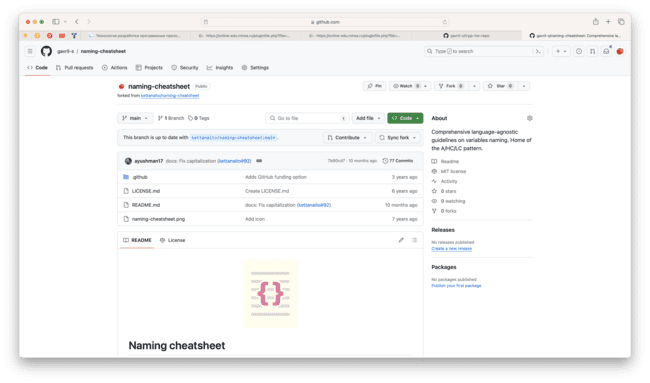


Рисунок 47 – Форк репозитория

Клонируем репозиторий и создадим в нём две новые ветки, в каждую из которых произведём по одному коммиту.

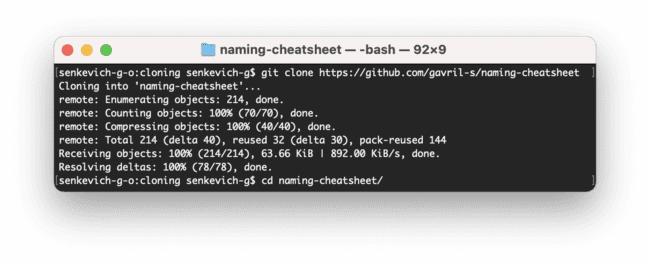


Рисунок 48 – Клонирование репозитория

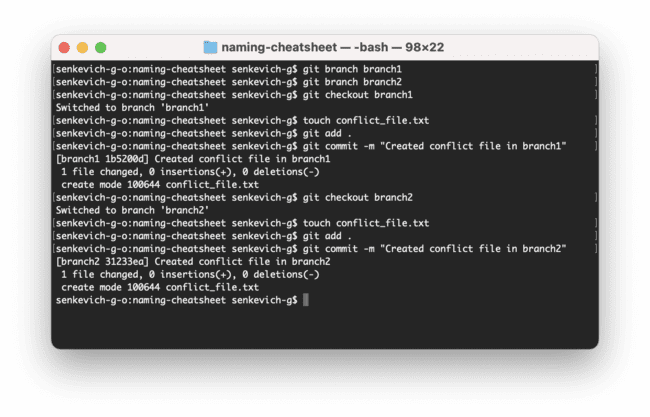


Рисунок 49 – Создание веток и коммиты

Сделаем ещё по 2 коммита в каждую ветку, которые изменяют один и тот же файл.

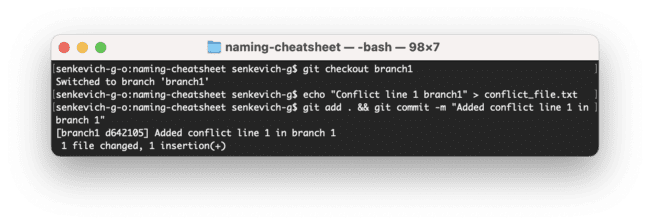


Рисунок 50 – Второй коммит в branch1

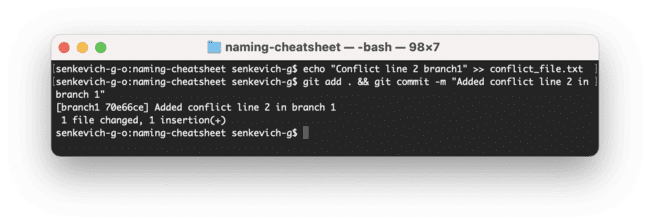


Рисунок 51 – Третий коммит в branch1

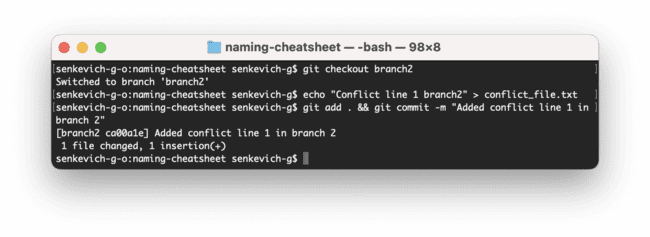


Рисунок 52 – Второй коммит в branch2

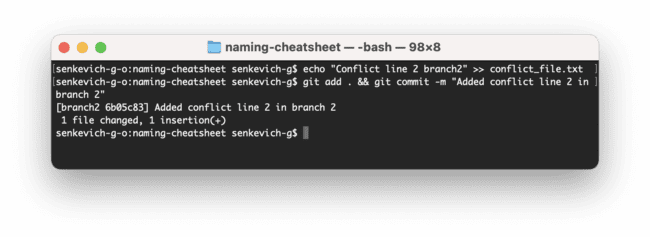


Рисунок 53 – Третий коммит в branch2

Выполним слияние веток, разрешив при этом конфликты.

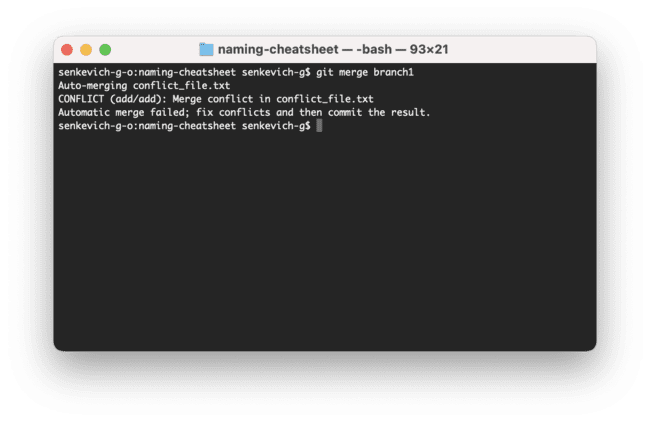


Рисунок 54 – Инициирование слияния

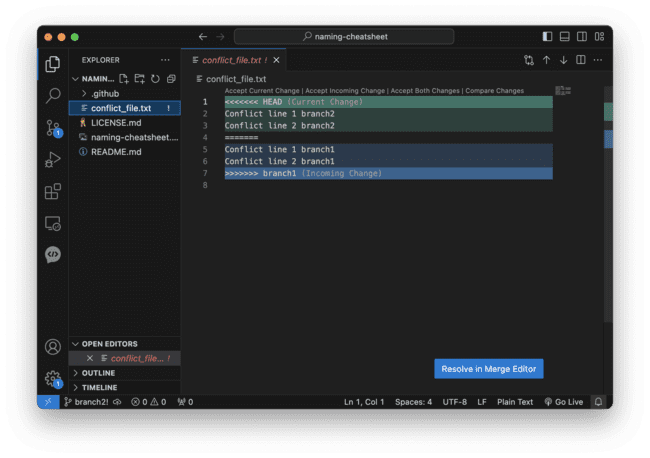


Рисунок 55 – Разрешение конфликтов

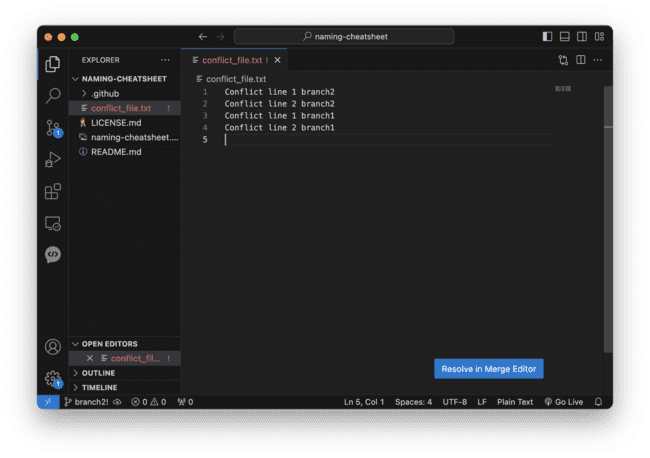


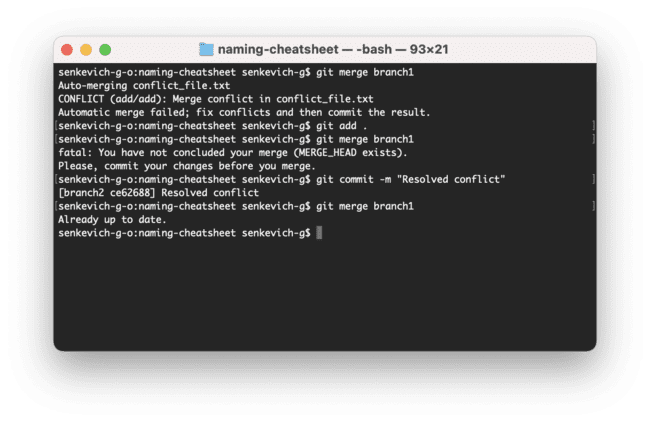
Рисунок 56 – Разрешение конфликтов

Рисунок 57 – Завершение слияния веток

Выгрузим изменения из всех веток в удалённый репозиторий.

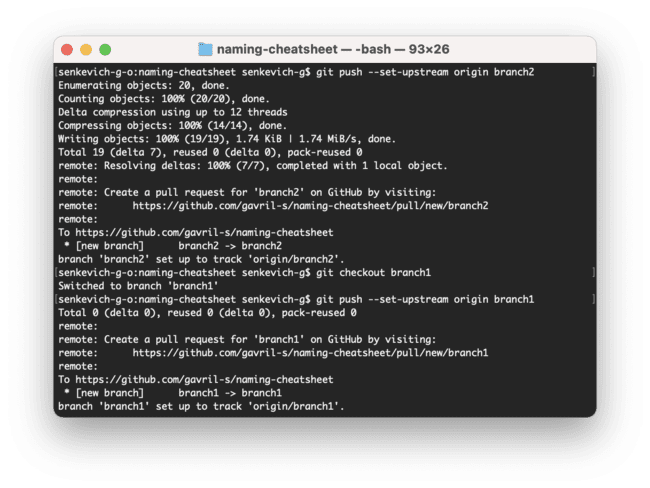


Рисунок 58 – Выгрузка изменений

Произведём ещё 3 коммита в ветку branch1.

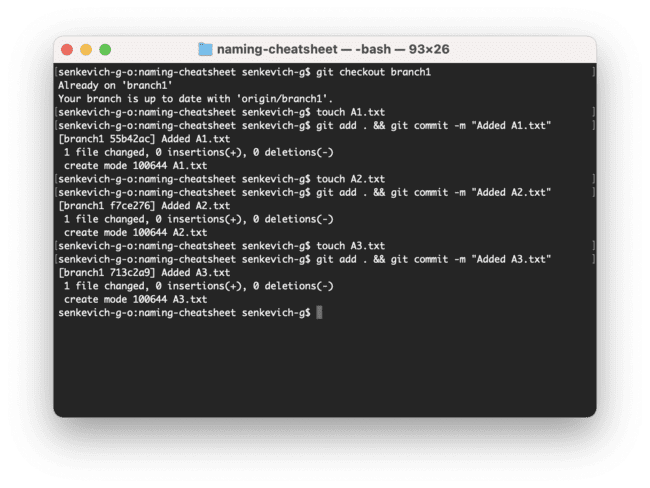


Рисунок 59 – Ещё 3 коммита в branch1

Клонируем репозиторий ещё раз в другую директорию, там произведём ещё 3 коммита в ветку branch1 и выгрузим изменения в удалённый репозиторий.

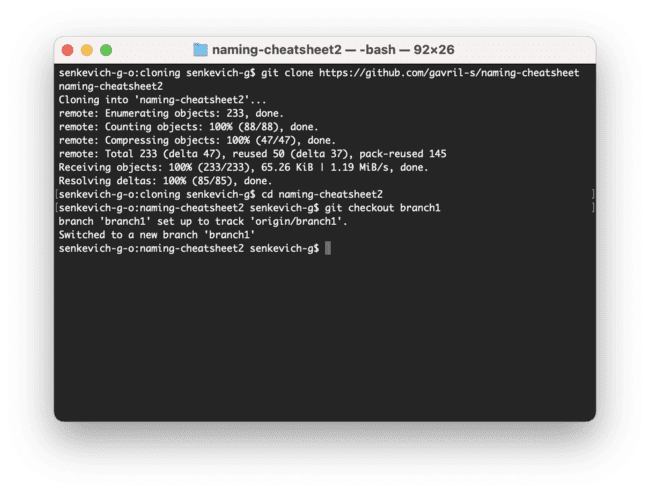


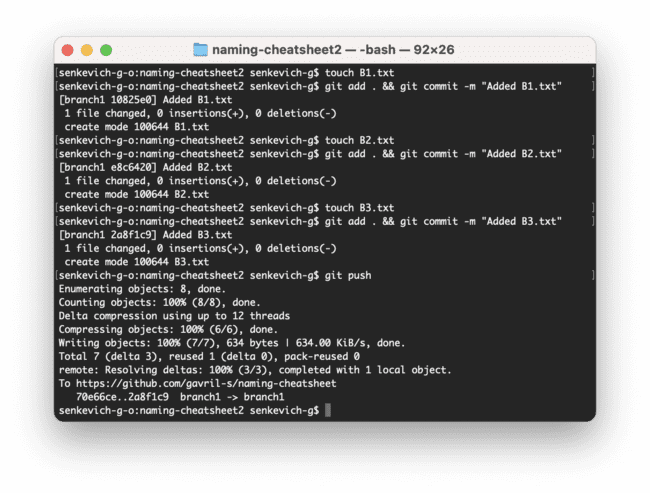
Рисунок 60 – Клонирование репозитория в другую директорию

Рисунок 61 – Ещё 3 коммита в ветку branch1 и git push

Вернёмся в старый клон репозитория и выгрузим изменения с опцией force, после чего вернёмся в новый клон и получим там все изменения.

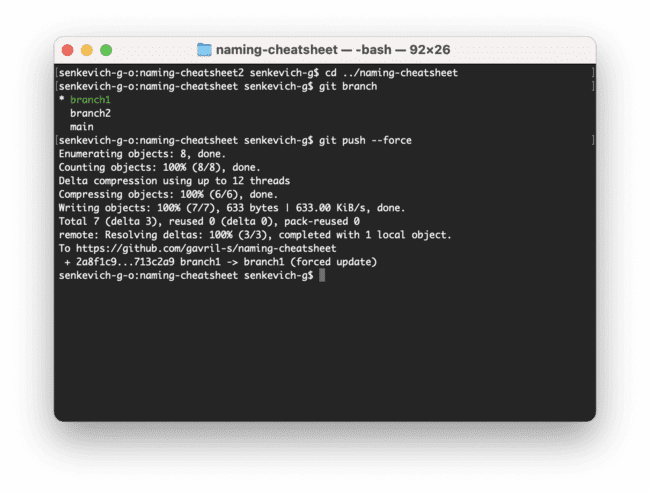


Рисунок 62 – Выгрузка изменений из старого клона

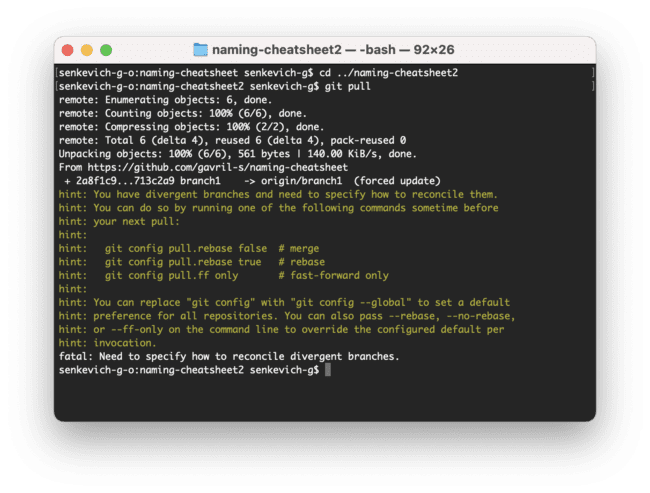


Рисунок 63 – Получение изменений в новом клоне

Ответы на вопросы

1. Для чего нужен .gitignore?

Файл .gitignore используется для указания Git'у, какие файлы и директории должны быть проигнорированы при отслеживании изменений.

2. Что делает команда git status?

Команда git status показывает текущий статус репозитория, включая измененные, добавленные или неотслеживаемые файлы.

3. Что делает команда git add?

Команда git add добавляет изменения в рабочей директории в индекс, подготавливая их к коммиту.

4. Что делает команда git log?

Команда git log выводит историю коммитов с информацией об авторе, дате и сообщении коммита.

5. Что делает команда git diff?

Команда git diff показывает различия между рабочей директорией и индексом, а также между коммитами.

6. Что делает команда git show?

Команда git show показывает информацию о конкретном коммите, включая изменения, внесенные этим коммитом.

7. Что делает команда git stash?

Команда git stash временно сохраняет изменения, позволяя переключаться между ветками без фиксации изменений.

8. Что сделает команда "git branch" без какого-либо параметра?

Команда git branch без параметров выводит список веток, а текущую ветку отмечает звездочкой.

Вывод

В результате выполнения работы были получены навыки по работе с командной строкой и Git.