**Лабораторная работа 3**

**Вариант 9.**

**Использовать в работе 4 разных файла.**

1. **Первоначальная настройка *git*.**

На данном этапе мы указали свое имя и электронную почту, а также проверили список нашей конфигурации (Рисунок 1).

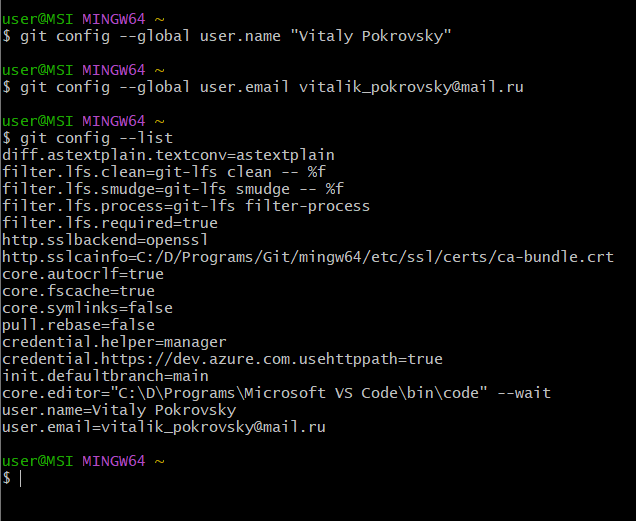


Рисунок 1 - первичная настройка git

1. **Создание проекта и репозитория**

На данном этапе мы создаем новый каталог, в котором будем работать. Результат выполненной работы представлен на рисунках 2 – 6.

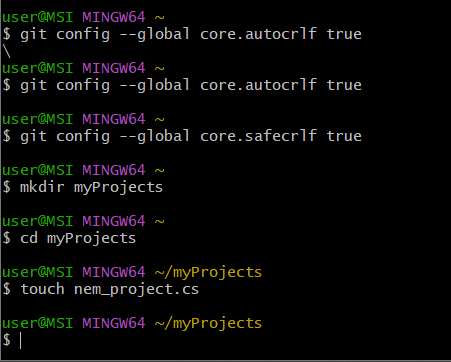


Рисунок 2 - создание каталога и проекта внутри него

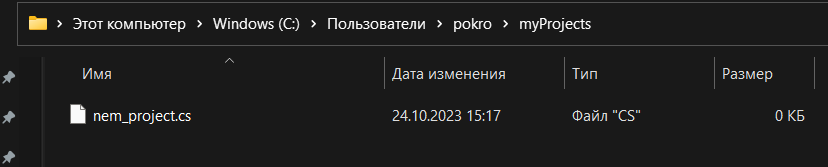


Рисунок 3 - результат создания каталога и проекта внутри него

Далее создаем репозиторий.

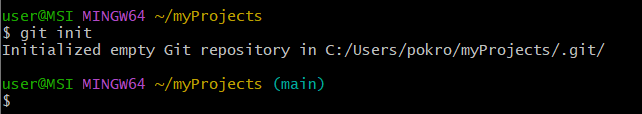


Рисунок 4 - создание репозитория

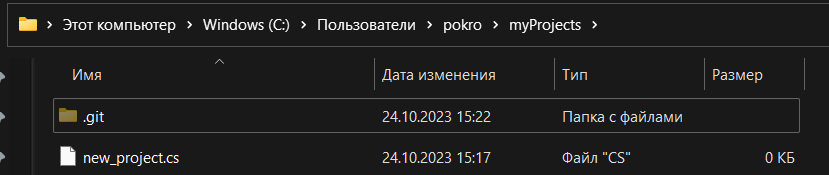


Рисунок 5 - результат создания репозитория (переименовали проект из-за опечатки)

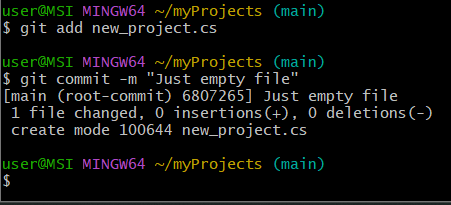


Рисунок 6 - Добавление проекта в репозиторий

1. **Проверка состояния репозитория**

Проверяем состояние репозитория для дальнейшей работы с ним (рисунок 7).

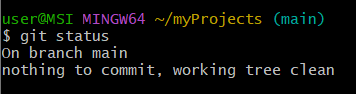


Рисунок 7 – Проверка состояния репозитория

1. **Индексация изменений**

Для сохранения версий, необходимо использовать коммиты. По сути, мы индексируем каждую новую версию проекта (рисунок 8 – 9).

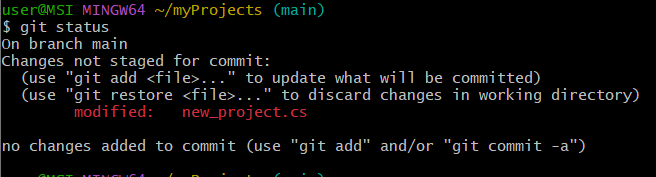


Рисунок 8 - проверка состояния после внесения изменений в проект

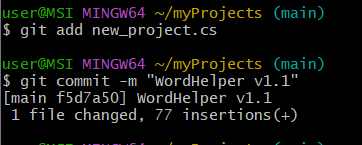


Рисунок 9 - Добавление нового коммита

1. **История версий**

История версий необходима для мониторинга наших версий (рисунок 10 – 12)

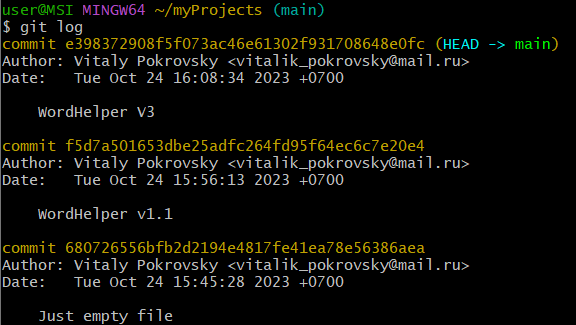


Рисунок 10 - История версий

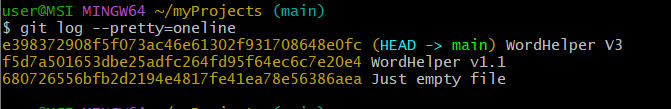


Рисунок 11 - История версий в однострочном формате

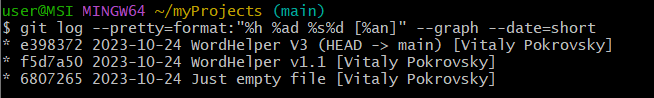


Рисунок 12 - История версий в своем формате

1. **Алиасы**

Алиасы (ассоциации) необходимы нам для упрощения ввода команд (рисунок 13 – 17).

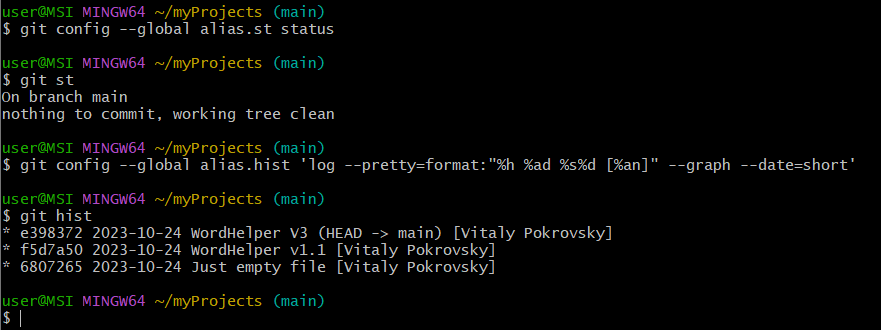


Рисунок 13 - Добавляем алиасы для более удобного ввода команд

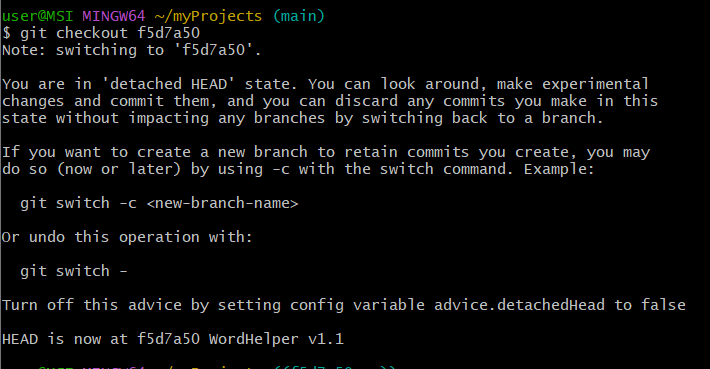


Рисунок 14 - Возврат к ранним версиям

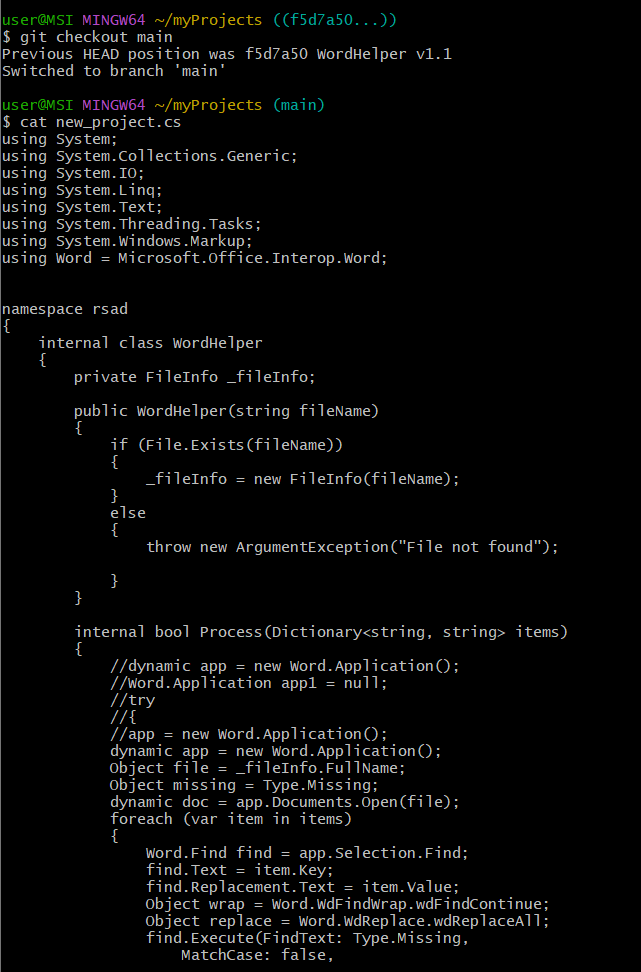


Рисунок 15 - Просмотр содержимого файла ч1.

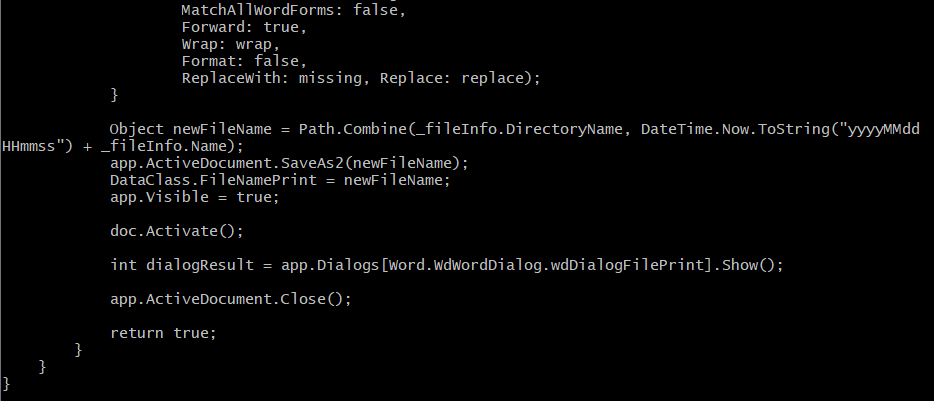


Рисунок 16 - просмотр содержимого файла ч.2

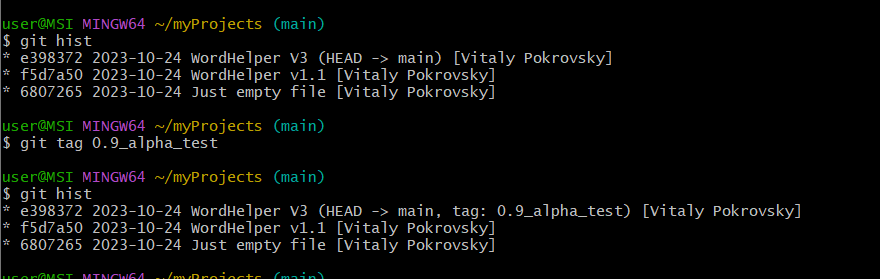


Рисунок 17 - Добавляем теги

1. **Отмена локальных изменений**

Сохраняем файл с ошибкой. Теперь мы можем отменить изменения (рисунок 18 – 19).

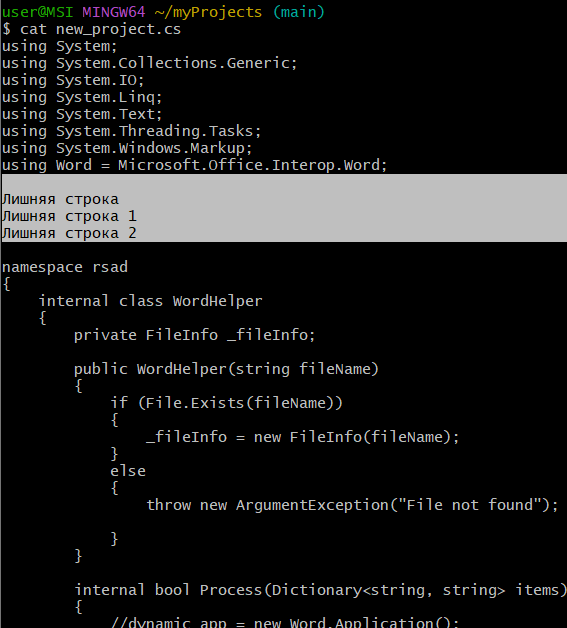


Рисунок 18 - Проект с ошибкой

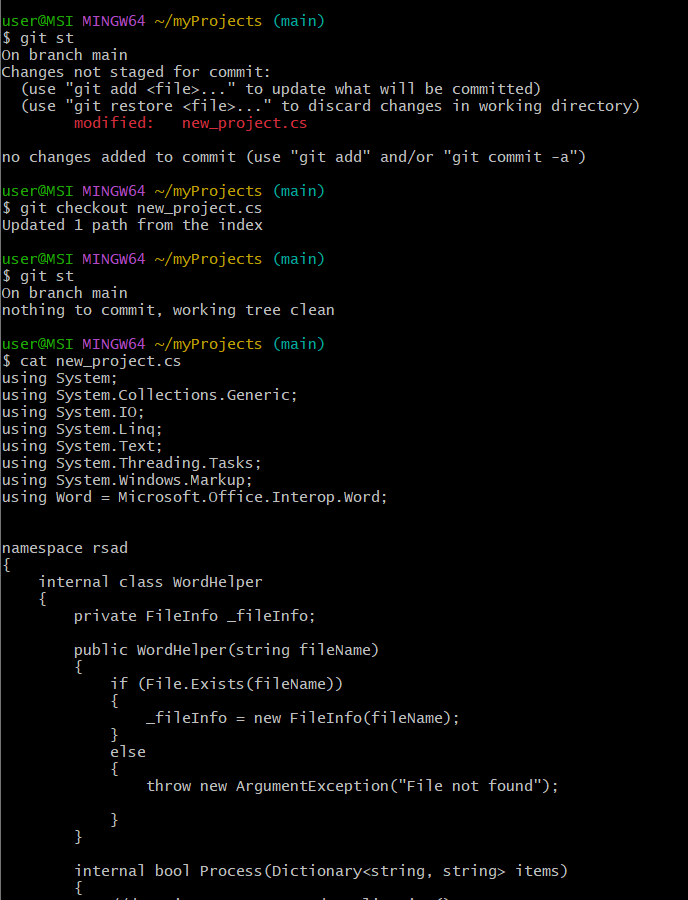


Рисунок 19 - Отмена локальных изменений

1. **Отмена проиндексированных изменений**

Бывает, когда в код были добавлены лишние строки и нам необходимо вернуться на прошлую версию проекта (рисунок 20 – 25).

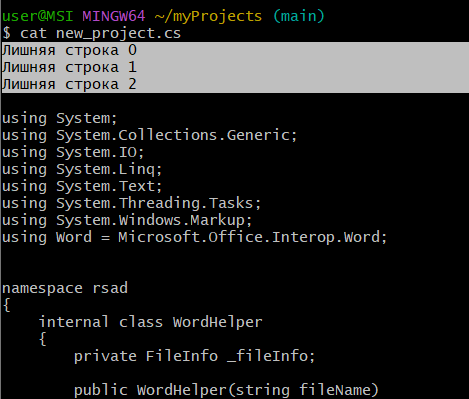


Рисунок 20 - Исходный файл с ошибкой

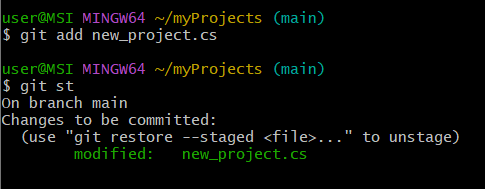


Рисунок 21 - Индексация изменений

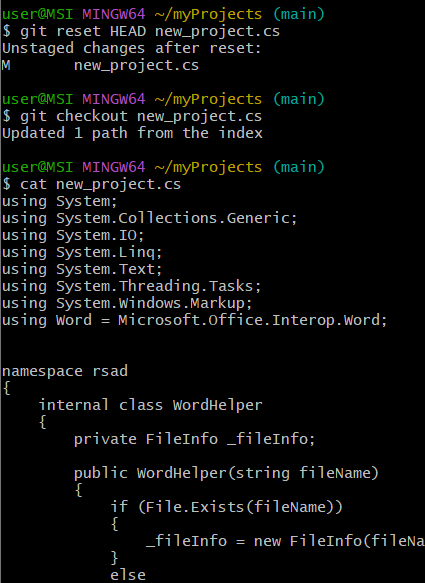


Рисунок 22 - Отмена изменений

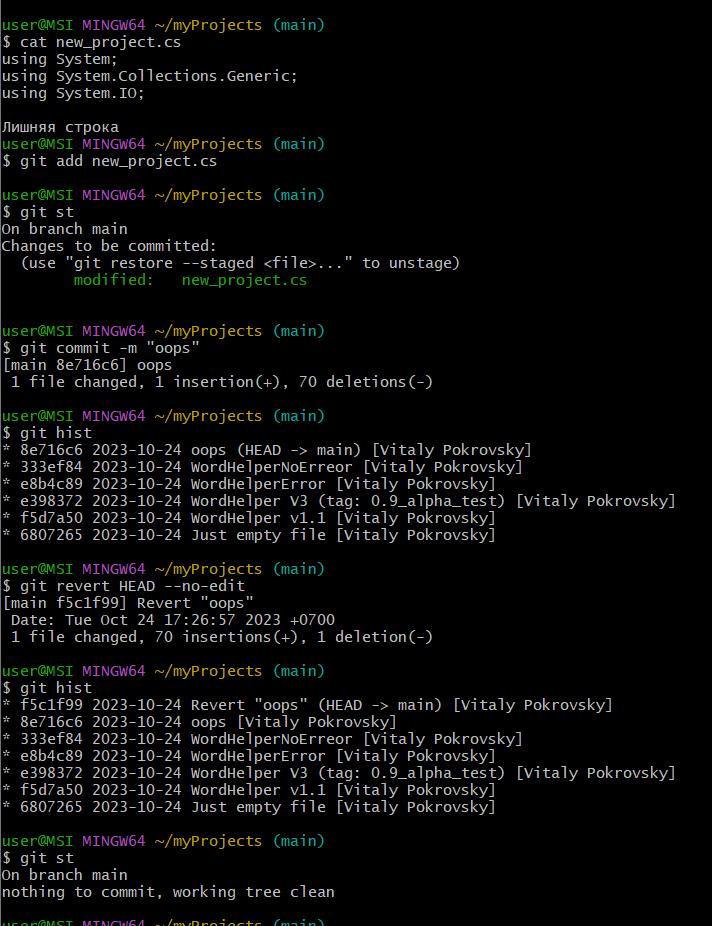


Рисунок 23 - откат изменений с примененным коммитом

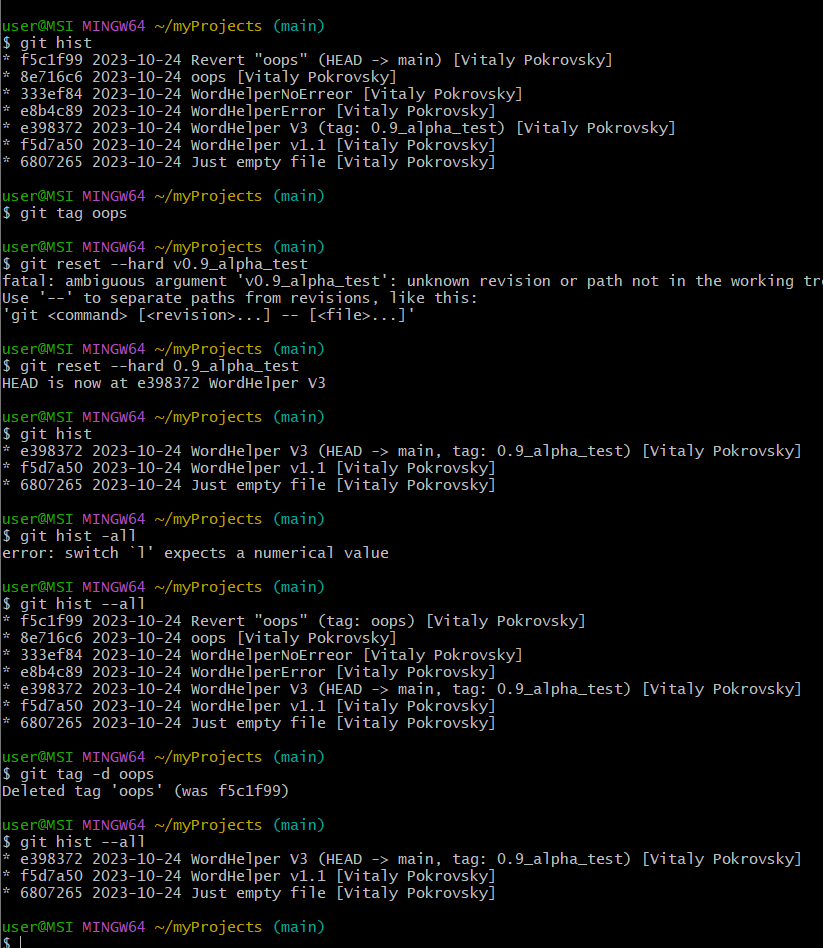


Рисунок 24 - Удаление коммита из истории

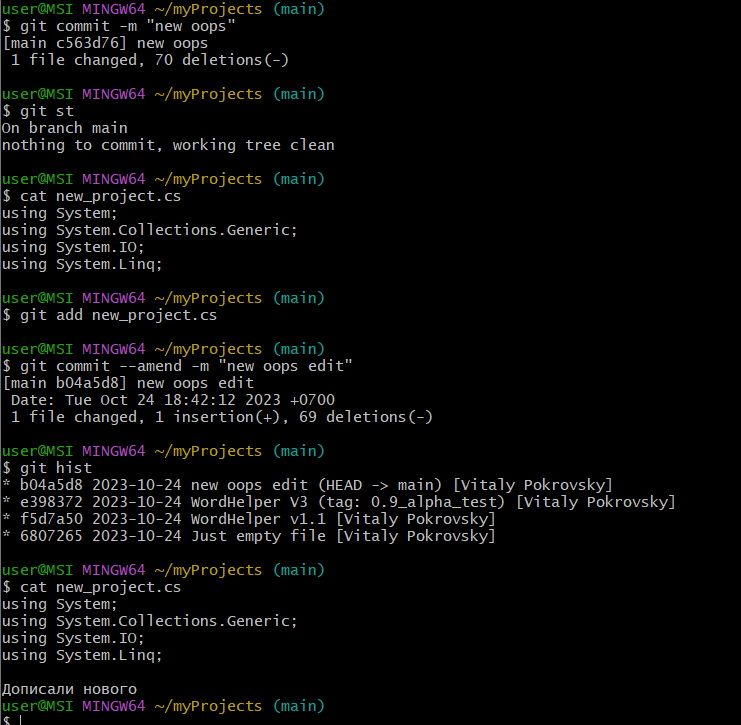


Рисунок 25 - Редактирование коммита

1. **Создание ветки**

Ветки необходимы для того, чтобы проект мог развиваться в разных направлениях, реализуя различные задачи (рисунок 26 – 29).

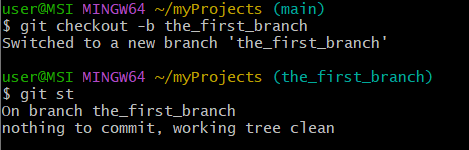


Рисунок 26 - Создание ветки

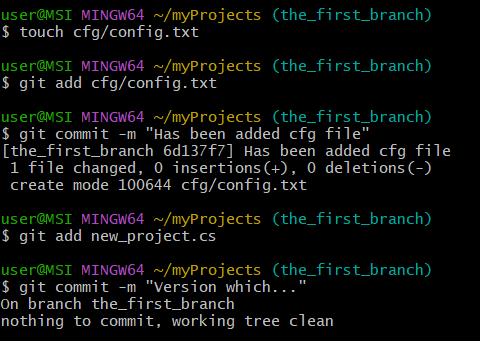


Рисунок 27 - Работа с ветками ч.1

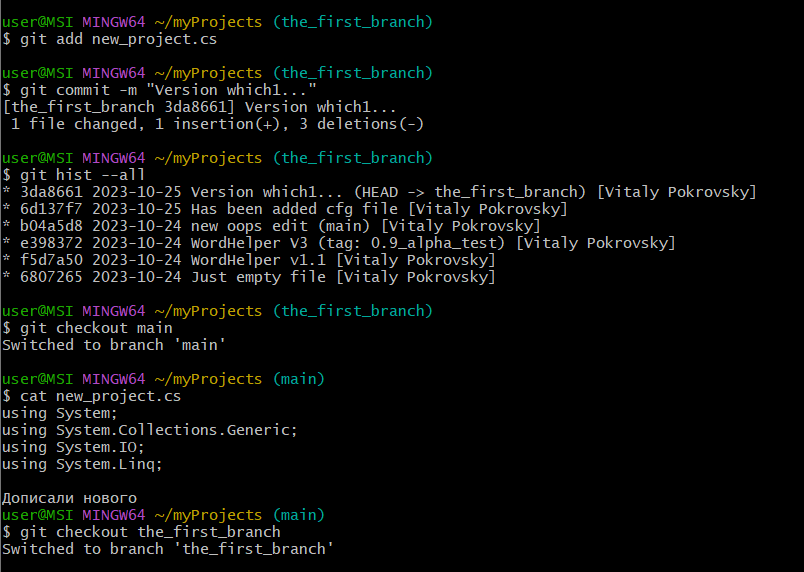


Рисунок 28 - Работа с ветками ч.2

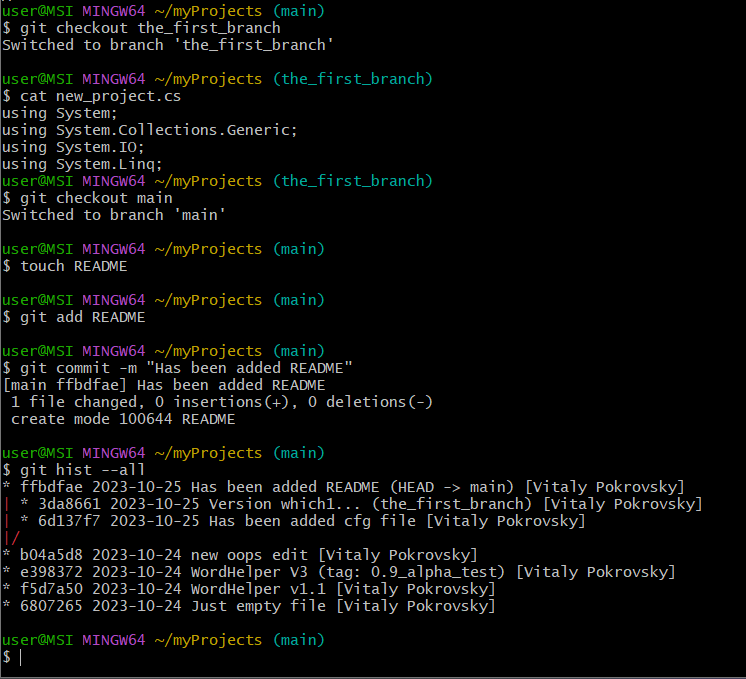


Рисунок 29 - Работа с ветками ч.3

1. **Слияние веток**

Объединяем ветки (рисунок 30).

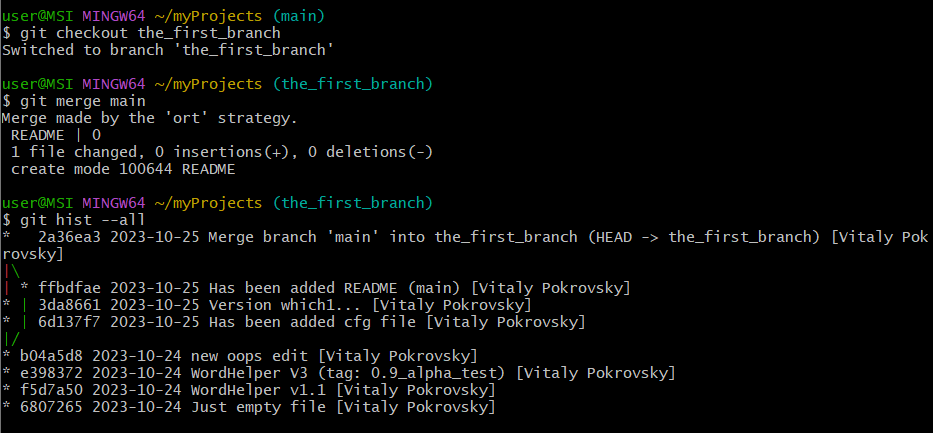


Рисунок 30 - Слияние веток

1. **Конфликт**

Система покажет нам конфликтные ситуации, если они имеются (рисунок 31 – 33).

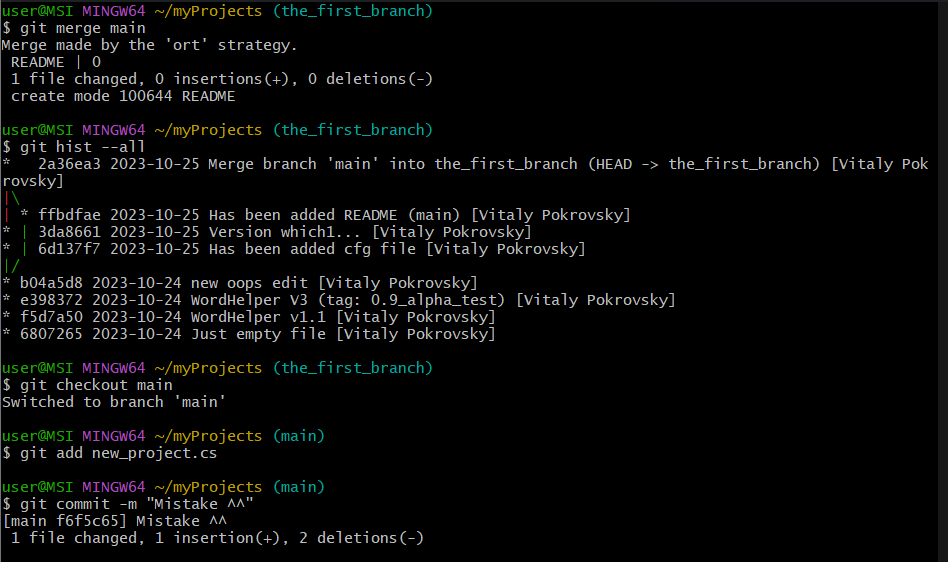


Рисунок 31 - конфликт ч.1

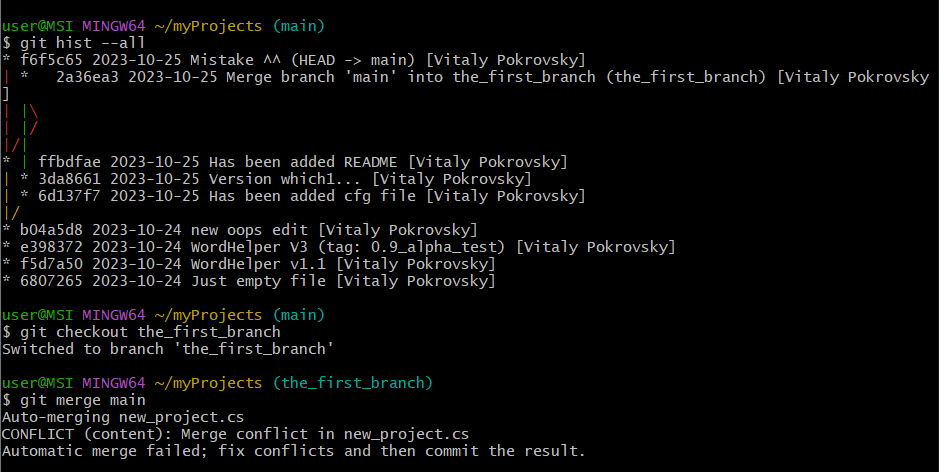


Рисунок 32 - Конфликт ч.2

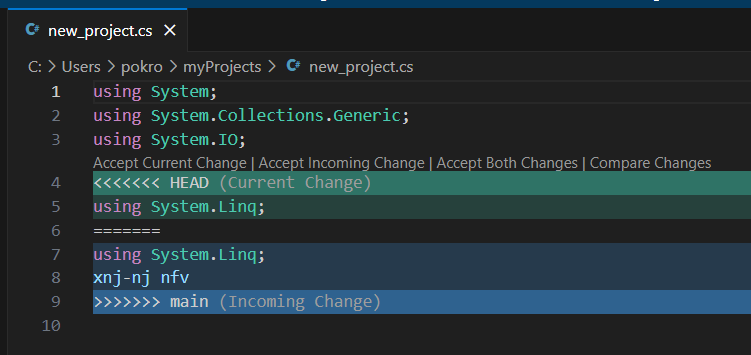


Рисунок 33 - Подсказка от git

1. **Разрешение конфликтов**

Исправляем конфликты (рисунок 34).

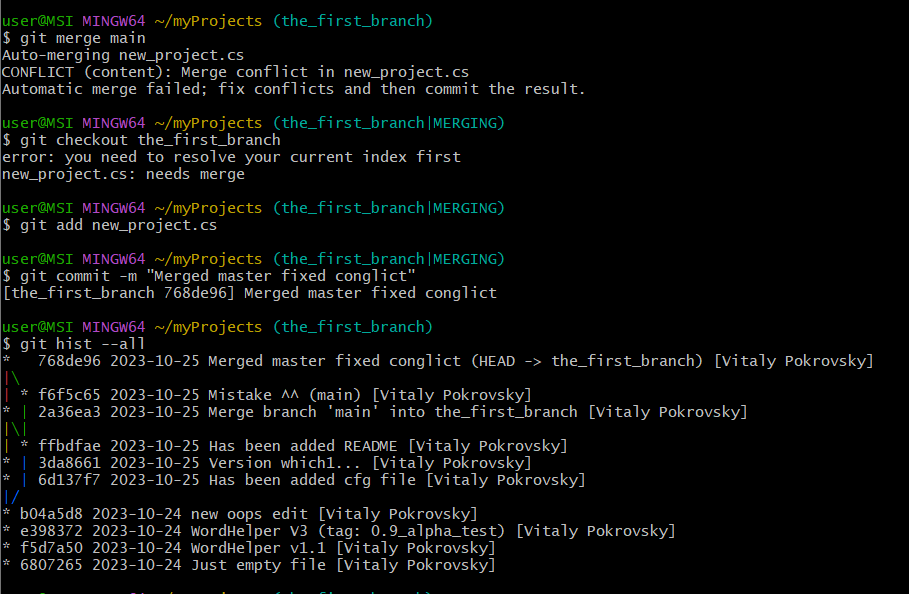


Рисунок 34 - Разрешение конфликта

1. **Создание чистого репозитория**

Обычный git-репозиторий подразумевает, что вы будете использовать его как рабочую директорию, поэтому вместе с файлами проекта в актуальной версии, git хранит все служебные, «чисто-репозиториевские» файлы в поддиректории .git. В удаленных репозиториях нет смысла хранить рабочие файлы на диске (как это делается в рабочих копиях), а все что им действительно нужно – это дельты изменений и другие бинарные данные репозитория. Вот это и есть «чистый репозиторий»

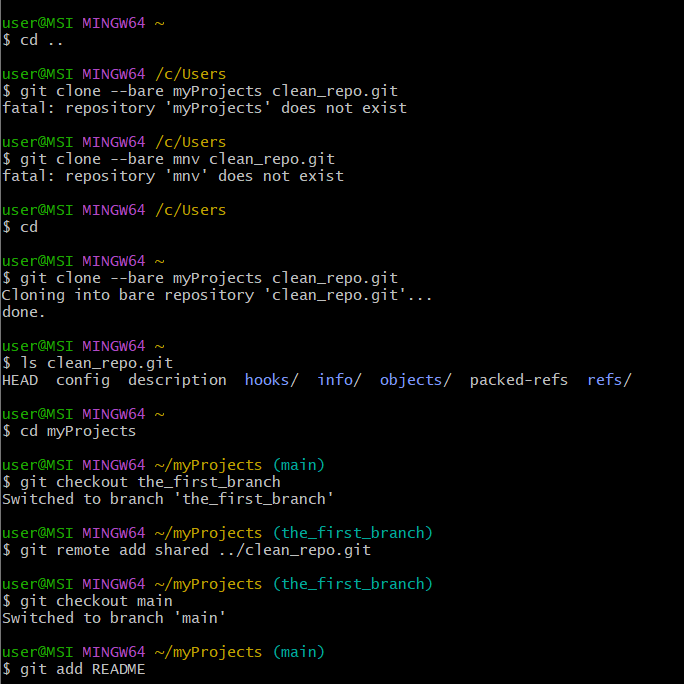


Рисунок 35 - Создание репозитория clean\_repo ч.1

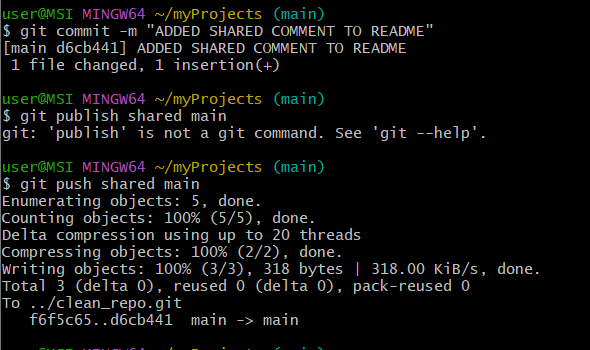


Рисунок 36 - Создание репозитория clean\_repo ч.2

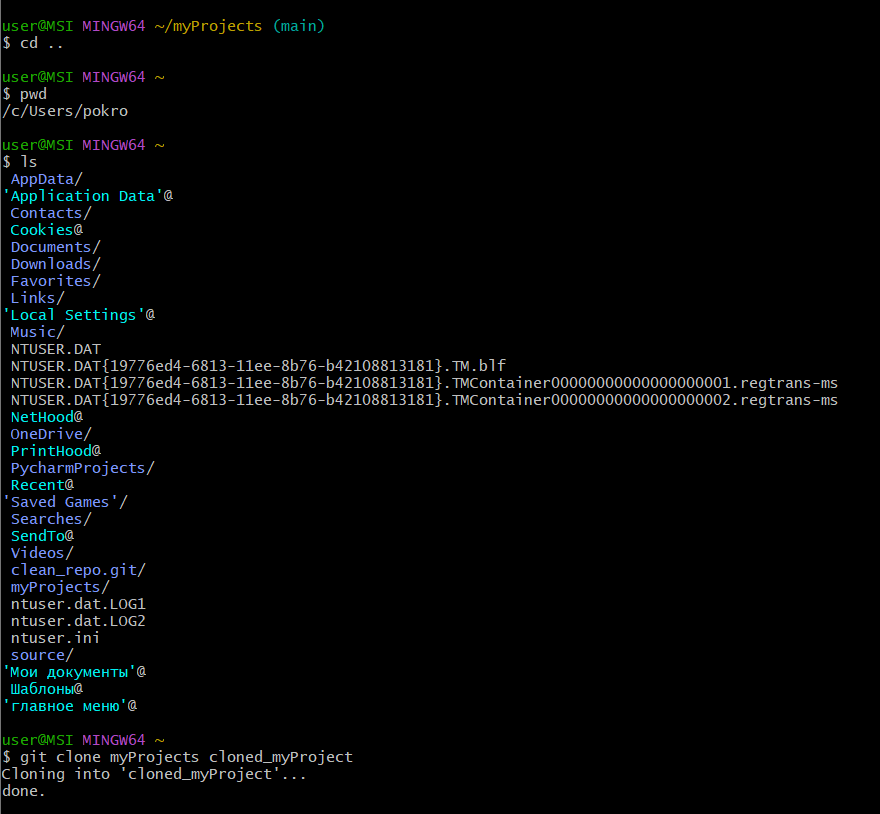


Рисунок 37 - Работа с удаленным репозиторием ч.1

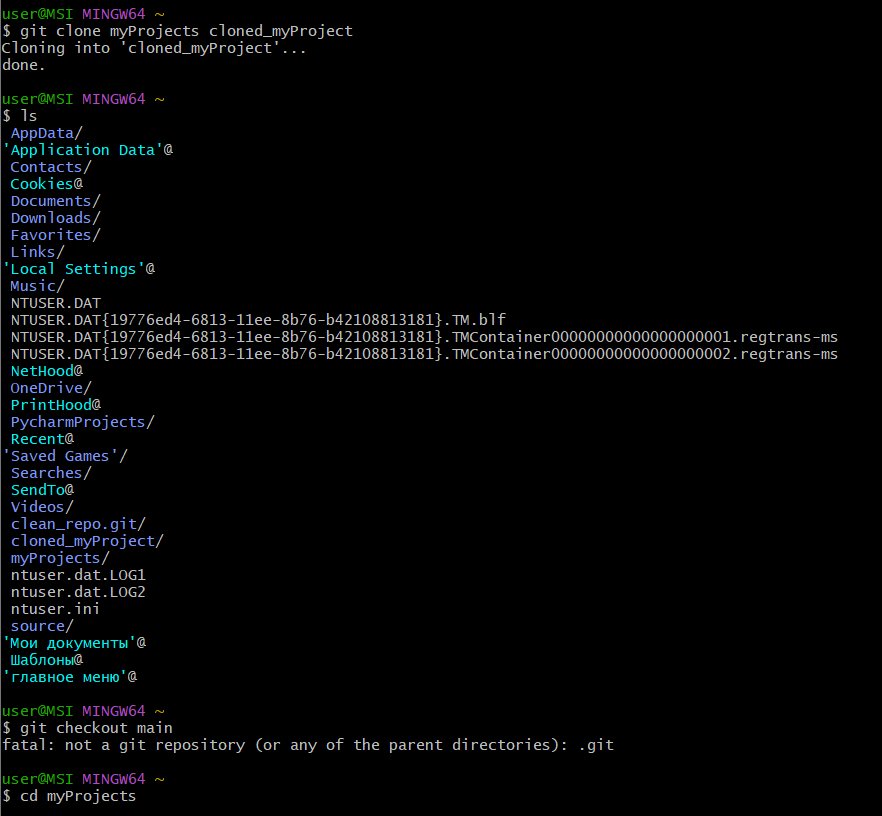


Рисунок 38 - Работа с удаленным репозиторием ч.2

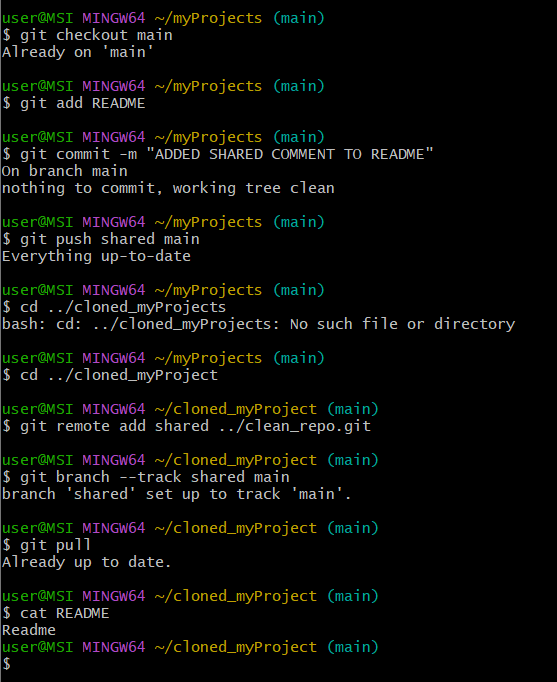


Рисунок 39 - Работа с удаленным репозиторием ч.3

**Работа по варианту 9. Использовать 4 разных файла.**

**Работа с первым файлом WordHelper.cs.**

Создаем файл, далее наполняем его кодом. Так же создаем версии для нашего файла. Результат представлен на рисунках 40-41.

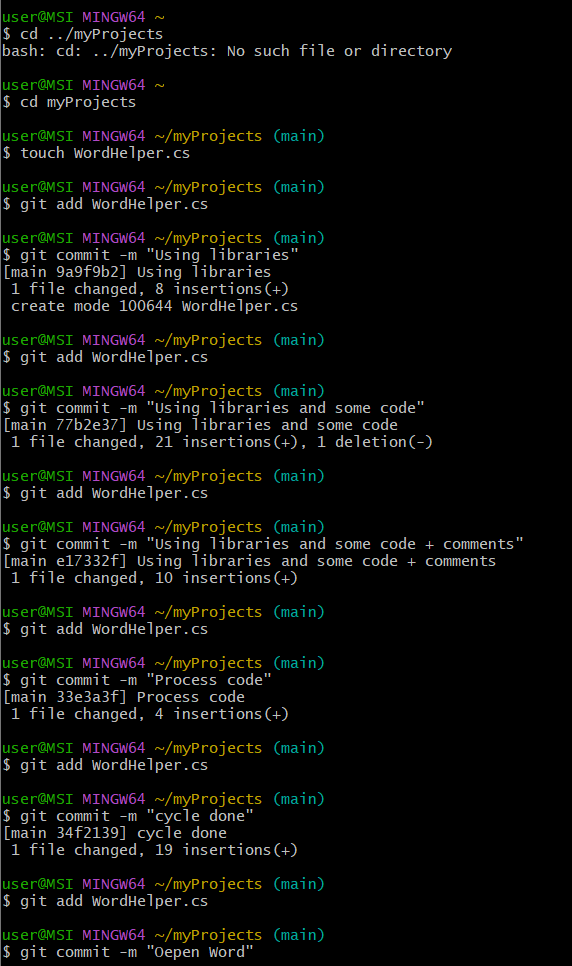


Рисунок 40 - Работа с первым файлом WordHelper.cs ч.1

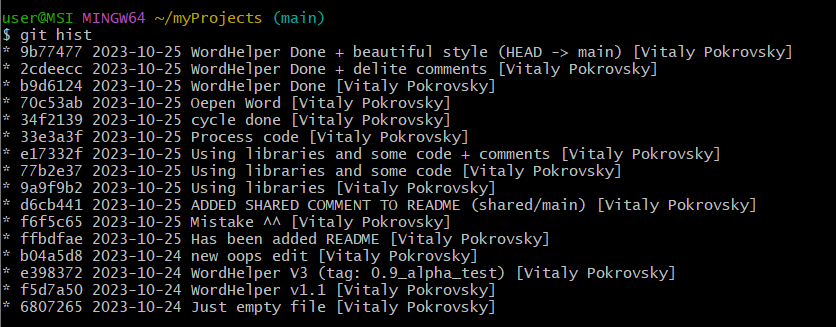


Рисунок 41 - Работа с первым файлом WordHelper.cs ч.2

**Работа со вторым файлом DB.cs**

Создаем файл, далее наполняем его кодом. Так же создаем версии для нашего файла. Результат проделанной работы представлен на рисунке 42.

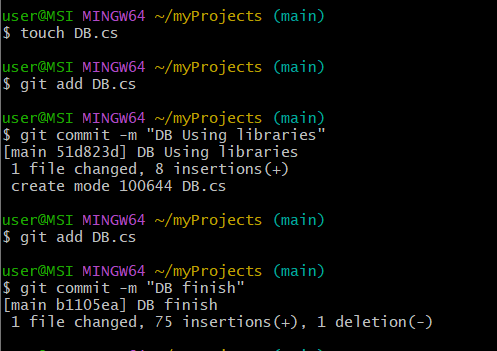


Рисунок 42 - Работа с файлом DB.cs

**Работа с третьим файлом AdminPanel.xaml.cs**

Создаем файл, далее наполняем его кодом. Так же создаем версии для нашего файла. Результат проделанной работы представлен на рисунке 43.

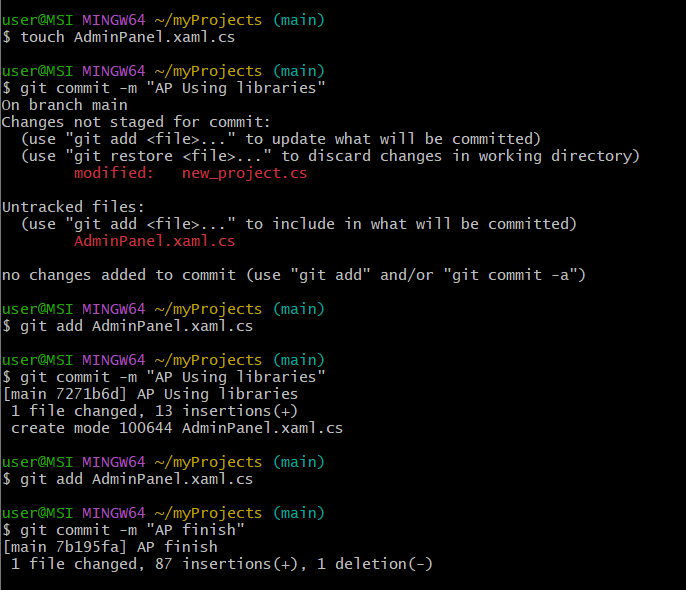


Рисунок 43 - Работа с файлом AdminPanel.xaml.cs

**Работа с четвертым файлом RequestWindow.xaml.cs**

Создаем файл, далее наполняем его кодом. Так же создаем версии для нашего файла. Результат проделанной работы представлен на рисунке 44.

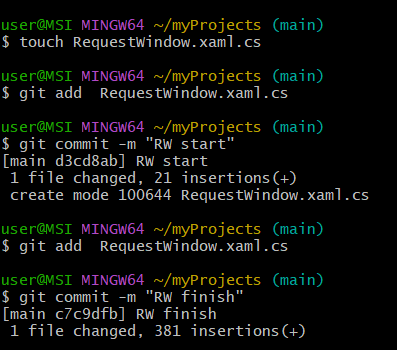


Рисунок 44 - Работа с файлом RequestWindow.xaml.cs

Посмотрим историю и наполнение файлов (рисунок 45-46).

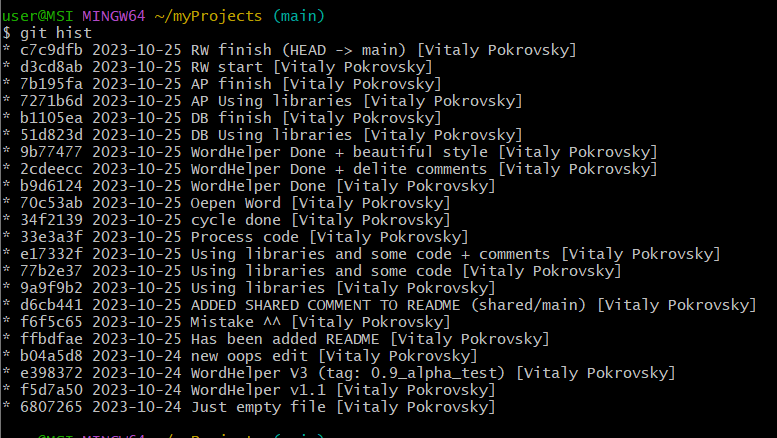


Рисунок 45 - История версий

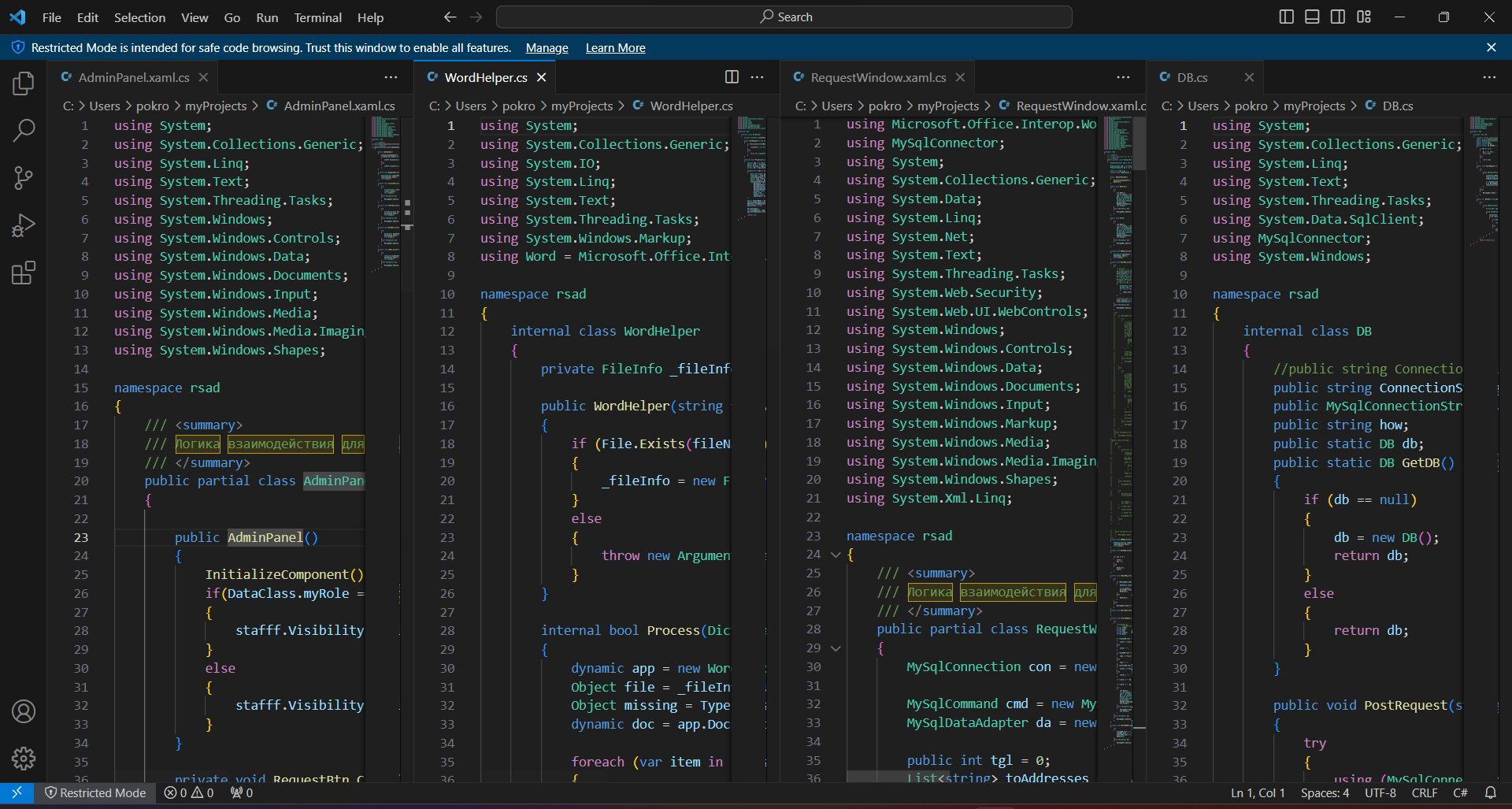


Рисунок 46 - Наполнение файлов

**Дополнительное задание**

Загрузили свой проект на github.com (рисунок 47-48).

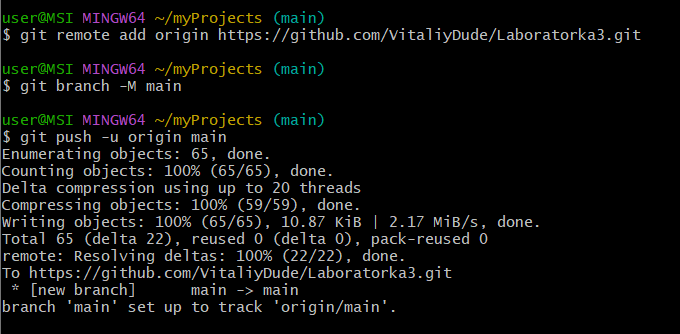


Рисунок 47 - Загрузка проекта на GitHub