Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

**дисциплины**

**«Информационные технологии и программирование»**

**Вариант**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Маликов Максим Егорович  1 курс, группа ИНС-б-о-25-1  09.03.02 «Информационные технологии и сети», направленность (профиль) «Прикладное программирование в интеллектуальных информационных системах», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Проверила:  Ассистент департамента цифровых, робототехнических систем и электроники Хацукова А.И.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2025 г.

**Тема:** Git локально: Работа с GitHub Desktop

**Цель: Научится работать с Git Dekstop.**

**Порядок выполнения работы:**

**Теория**

Система контроля версий (Version Control System - VCS) — это программное обеспечение, которое помогает отслеживать изменения в файлах и координировать работу нескольких людей над общими файлами.

Основные виды *VCS*:

* Локальные — версии хранятся на локальном компьютере (*RCS*, *SCCS*)
* Централизованные — единый сервер хранит все версии (*SVN*, *CVS*)
* Распределенные — каждый участник имеет полную копию репозитория (*Git*, *Mercurial*)

Ключевые понятия:

* Репозиторий — хранилище всех версий файлов и метаданных
* Коммит — снимок состояния файлов в определенный момент времени
* Ветка — указатель на определенный коммит
* Рабочая область — текущее состояние файлов
* Область подготовки (Staging Area) — промежуточная область между рабочей областью и репозиторием

Три состояния файлов в Git:

1. Modified — измененный в рабочей области
2. Staged — подготовленный к коммиту
3. Committed — сохраненный в репозитории

GitHub Desktop — это кроссплатформенное приложение, которое предоставляет визуальный интерфейс для работы с Git. Оно абстрагирует сложные командные операции, делая работу с системой контроля версий более доступной.

*Преимущества GitHub Desktop:*

* Визуальное представление истории коммитов
* Упрощенное управление ветками
* Интеграция с платформой GitHub
* Визуальное разрешение конфликтов
* Поддержка больших файлов через Git LFS

GitHub — это платформа для хостинга Git-репозиториев с дополнительными функциями:

* Pull Requests для code review
* Issues для отслеживания задач
* Actions для CI/CD
* Wiki для документации
* Projects для управления проектами

Протоколы доступа:

* HTTPS — простая настройка, требует аутентификации
* SSH — более безопасный, использует ключи

Структура коммитов:

* Одно логическое изменение в одном коммите
* Четкие сообщения коммитов
* Следование *conventional commits*

Безопасность:

* Никогда не коммитьте чувствительные данные
* Используйте SSH ключи вместо паролей
* Включайте двухфакторную аутентификацию
* Регулярно обновляйте клиенты Git

Типичный workflow (рабочий процесс):

1. Получить последние изменения: git pull
2. Создать feature ветку: git checkout -b feature/name
3. Внести изменения и протестировать
4. Добавить файлы в staging: git add .
5. Создать коммит: git commit -m "message"
6. Отправить изменения: git push origin feature/name
7. Создать Pull Request на GitHub
8. Пройти code review
9. Слить изменения в main

Типы конфликтов:

* Конфликты слияния при merge
* Конфликты перебазирования при rebase
* Конфликты cherry-pick

Стратегии разрешения:

* Ручное редактирование конфликтующих файлов
* Использование инструментов слияния (*meld*, *kdiff3*)
* Принятие текущих или входящих изменений

**Практическая часть**

**Задание 1. Подготовка**

Установите и настройте Git и Github Desktop. (Рисунок 1-4.)

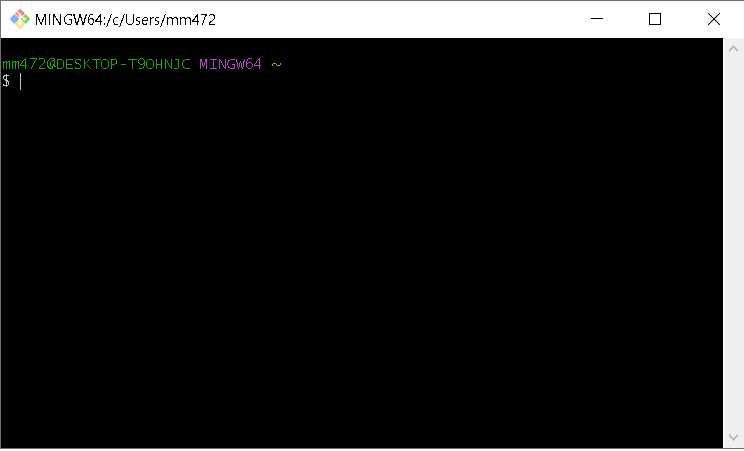


Рисунок №1. Установка GitBash

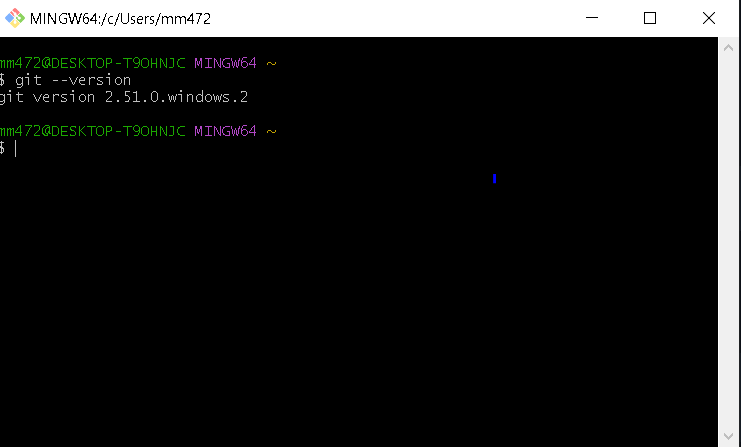


Рисунок №2. Проверка версии с помощью команды

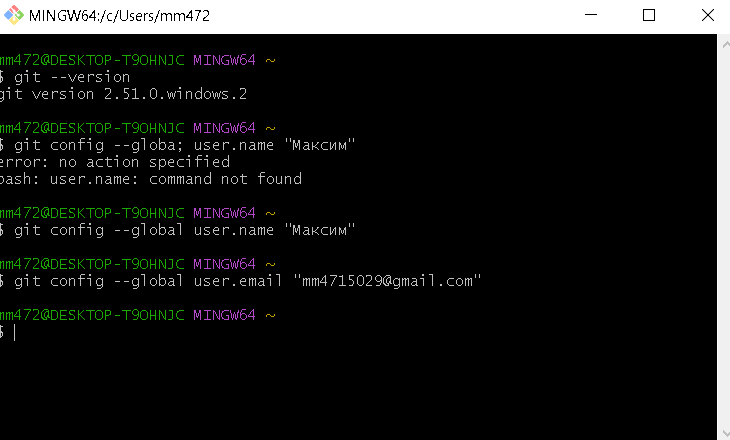


Рисунок №3. Проверка настроек на GitBash.

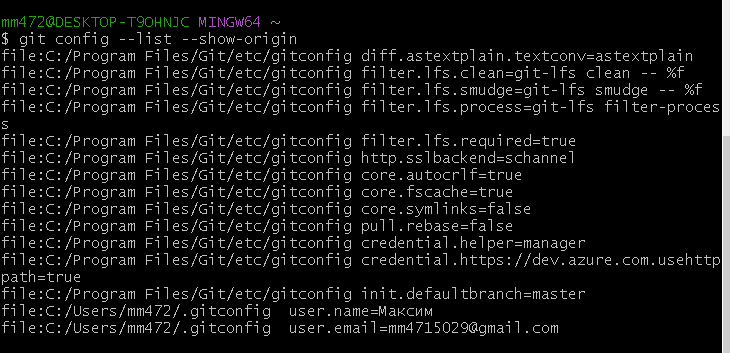


Рисунок №4. Итоги вывода команды всех настроек команды.

**Задание 2. Работа с Github Desktop**

Создайте или клонируйте репозиторий. (Рисунок 5-6).

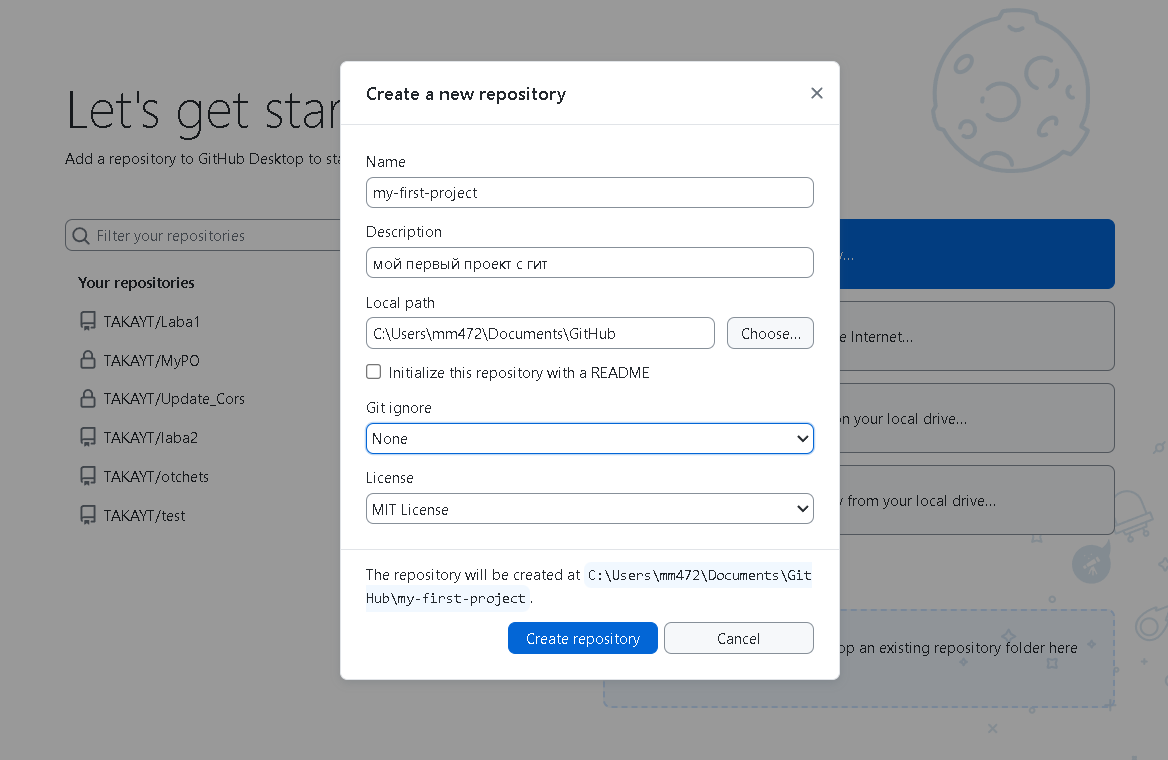


Рисунок №5. Создание первого репозитория в программе.

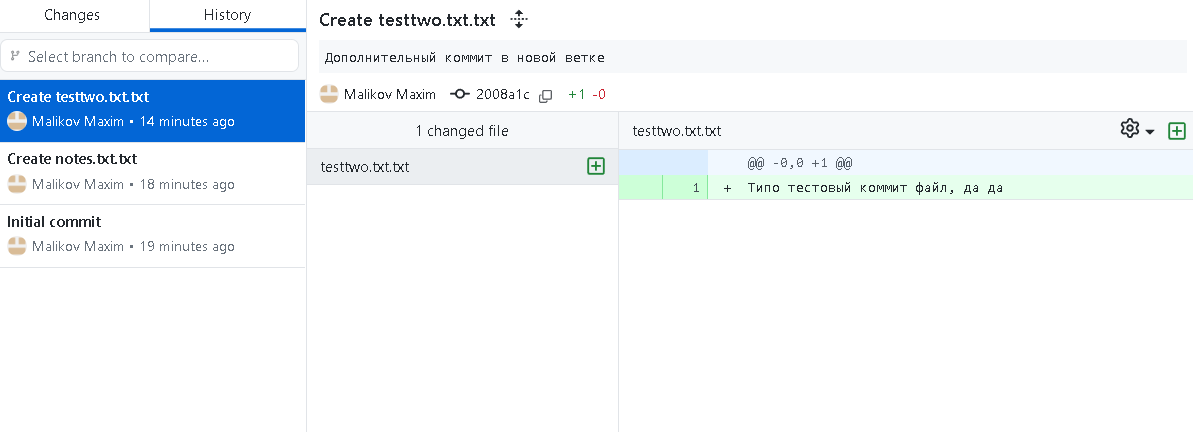


Рисунок №6. Проверка истории репозитория и его публикация в Weб версию сайта.

**Задание 3. Коммиты**

Создайте несколько коммитов (добавьте файлы, измените их), а затем запушьте (Push origin) в удалённый репозиторий. (Рисунок 7).

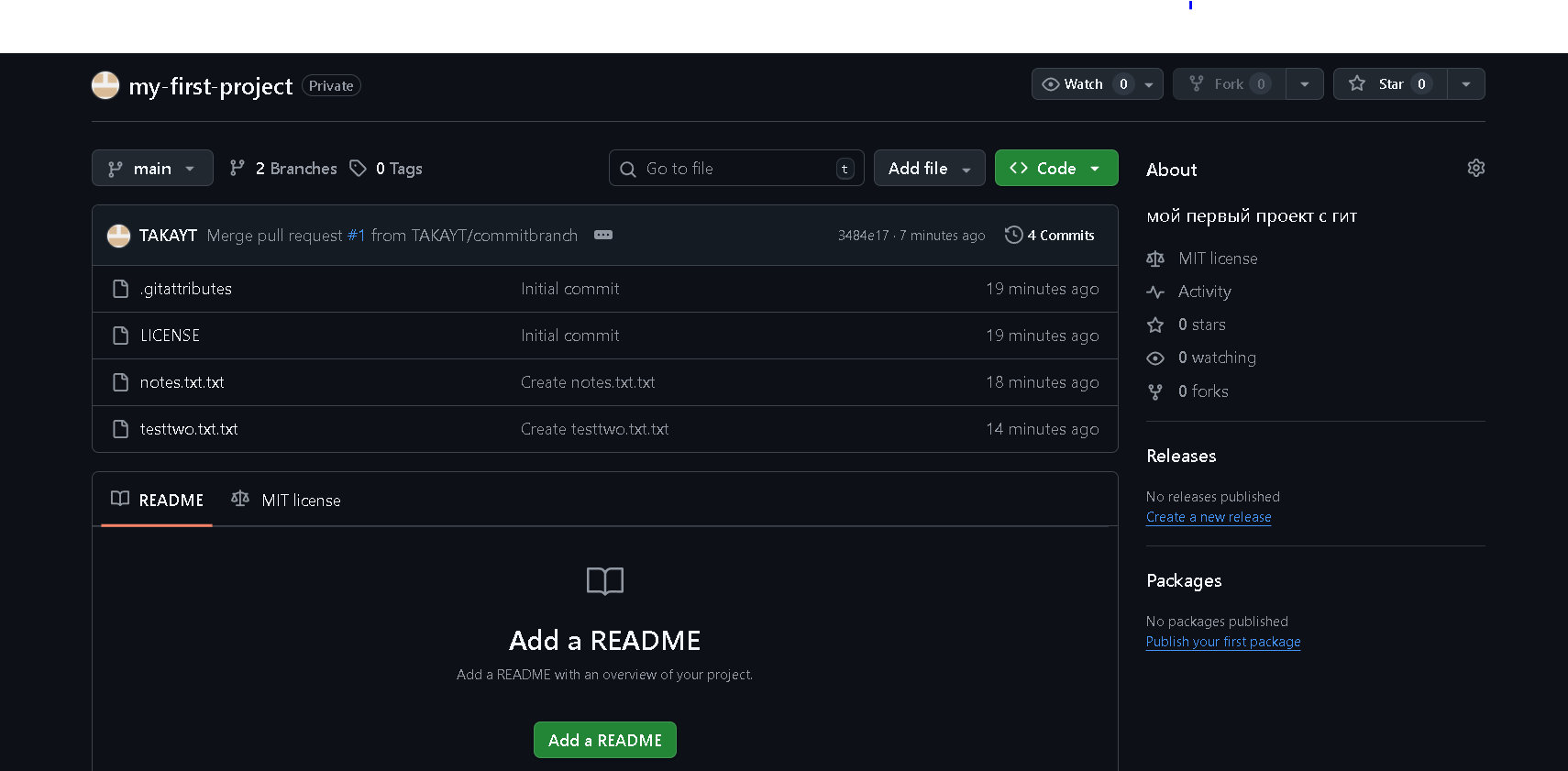


Рисунок №7. Итоговый вид репозитория

Контрольные вопросы

1. Для чего нужны команды git config --global user.name и git config --global user.email

Эти команды задают глобальные настройки пользователя Git: имя и электронную почту. Они используются для подписи коммитов и позволяют идентифицировать автора изменений во всех репозиториях на данном компьютере.

2. Как проверить, что Git установлен корректно

Выполните команду git --version. Если Git установлен, вы увидите номер версии. Также можно проверить доступность базовых команд, например, git status.

3. Как просмотреть историю изменений в репозитории

Используйте команду git log. Она показывает список всех коммитов с подробной информацией: автор, дата, сообщение. Для более краткого вывода можно использовать git log --oneline.

4. Что такое ветка в Git и для чего она используется

Ветка — это параллельная линия разработки, позволяющая изолировать изменения от основной версии кода. Ветки упрощают работу над новыми функциями, исправлением ошибок и эксперименты без риска повредить основную ветку.

5. Для чего нужен файл .gitignore

Файл .gitignore содержит список файлов и папок, которые Git не должен отслеживать. Это помогает исключить временные файлы, настройки среды разработки и другие лишние данные из коммитов.

Вывод: Мы научились работать в GitBash и переносить локальные репозитории в Web версию GitHub.