МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности

Кафедра компьютерных интеллектуальных технологий проектирования

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

По дисциплине: «Системы хранения и обработки данных»

Тема: «Основы работы с системой управления версиями Git»

Выполнил работу студент группы мИИВТ-241: Тогушов В.А.

подпись, дата

Принял: Короленко В.В.

подпись, дата

Воронеж 2024

Цель лабораторной работы:

изучить назначение распределённой системы управления версиями, освоить процесс создания репозитория и основы управления версиями файлов.

Основные задачи:

* создание профиля и репозитория на платформе github;
* установка git, запуск;
* создание ветки для отработки своего проекта;
* изучение основных команд git;
* клонирование удаленного репозитория на локальный компьютер;
* создание файла с данными о себе, изменение файла и отправка соответствующих изменений из локального репозитория на удаленный;
* откат изменений до предыдущего коммита.

Учебная задача:

1. Создать профиль на платформе github.
2. Создать репозиторий на платформе github.
3. Добавить комментарий в файл readme.md о создании репозитория, указать цель лабораторной работы, фамилии, имена, отчества участников команды вашего проекта. Репозиторий создавать один на команду (не более 4 человек).
4. Скачать и установить git на локальный компьютер (при необходимости). Запустить.
5. Ветка main (master) является основной веткой для вашего будущего проекта и воспринимается как своего рода «чистовик». Поэтому необходимо создать новую ветку для отработки своего проекта. Название ветки указывать в формате «LR4-FIO», где FIO – первые буквы фамилии, имени, отчества латиницей. Ветка создаётся для каждого участника команды. Дальнейшую работу проводить в своей ветке до момента слияния. Внимание! Новая ветка должна быть ответвлением от главной ветки! В противном случае операцию слияния провести будет невозможно.
6. Изучить основные команды git. Клонирование репозитория, инициализация, подготовка файлов к отправке, добавление коммита с комментарием (при выполнении задания комментарии для всех коммитов обязательны), отправка изменений на удаленный репозиторий, слияние, pull request и др. Описать изученные команды в файле «readme.md» в своей ветке.
7. Клонировать удаленный репозиторий на локальный компьютер. Дальнейшие изменения проводить в локальном репозитории.
8. Создать в локальном репозитории текстовый файл с названием «FIO.txt» (FIO – первые буквы фамилии, имени, отчества латиницей) с данными о себе. В файле указать следующие данные:

* фамилия, имя, отчество;
* операционная система, на которой вы работаете с Git;
* образование (специальность);
* место работы (если не работаете), то желаемую должность, организацию или направление будущей работы;
* ключевые навыки;
* ваши увлечения, хобби.

1. Сохранить изменения (необходимо работать в своей ветке) и отправить изменения из локального репозитория на удаленный с использованием изученных ранее команд. Не забывать про добавление исчерпывающих комментариев. Проконтролировать наличие изменений на удаленном репозитории.
2. Открыть на локальном компьютере созданный ранее файл «FIO.txt». Добавить данные о среднем балле по предыдущему образованию. Отправить изменения на удаленный репозиторий. Проконтролировать учёт изменений.
3. Открыть на локальном компьютере созданный ранее файл «FIO.txt». Добавить данные о месте рождения. Отправить изменения на удаленный репозиторий. Проконтролировать учёт изменений.
4. Произвести откат изменений до предыдущего коммита. Итогом должно стать такое состояние вашего репозитория в вашей ветке, каким оно было до внесения информации о месте рождения. Учтите, что удаление этой информации из файла и проведение нового коммита не является правильным решением данного пункта. Вам необходимо именно «откатиться» к состоянию предыдущего коммита.
5. Отправить ссылку на ваш репозиторий на платформе github на почту преподавателю. Репозиторий должен быть публичным.
6. При работе в команде необходимо выполнять работу на различных операционных системах. Приоритетный порядок выбора операционных систем: Astra Linux, Windows 10, CentOS, любая другая система, кроме указанных ранее.
7. У каждого участника команды должна быть своя ветка. Все ветки после выполнения задания необходимо будет интегрировать в главную ветку (при выполнении следующей лабораторной работы).
8. Подготовить отчёт о проделанной работе.

Индивидуальное задание:

Выполнить все пункты раздела «Учебная задача», оформить отчёт, представить результаты выполнения лабораторной работы к защите.

Первым делом для работы зайдем на свой аккаунт GitHub (уже был ранее зарегистрирован). Создадим новый репозиторий: введем его название, укажем тип доступа «Public» и поставим галочку для создания readme файла (Рисунок 1).

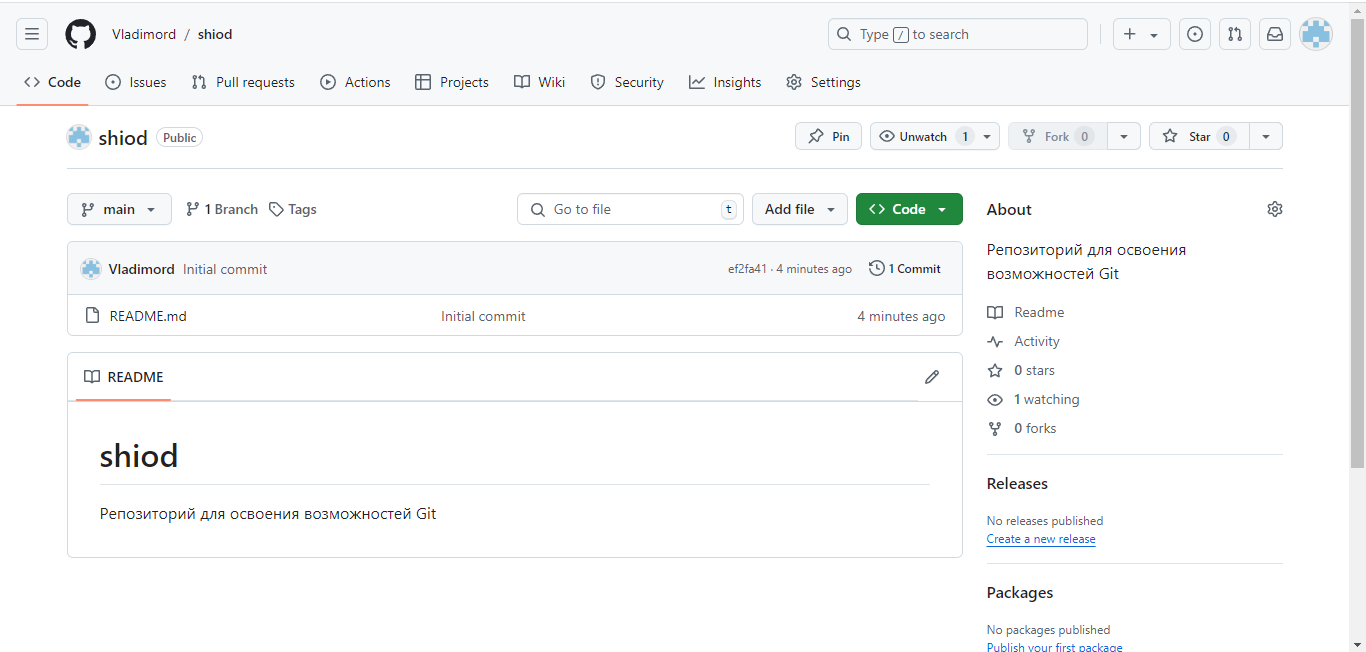


Рисунок 1 – Создание нового репозитория на GitHub

Теперь добавить комментарий в файл «readme.md» о созданном нами репозитории, также укажем в нем цель лабораторной работы, ФИО участников команды нашего проекта (Рисунок 2).

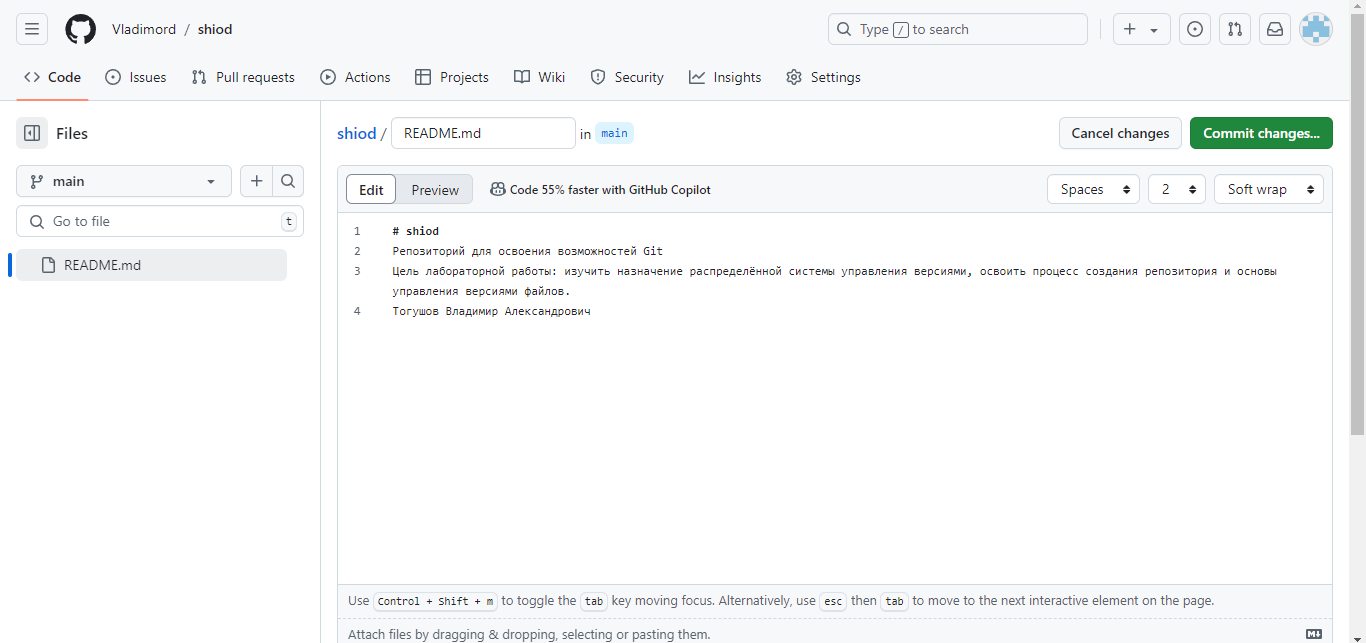


Рисунок 2 – Редактирование файла «readme.md»

Теперь скачаем и установим Git на Windows 11. Дистрибутив берем с официального источника, следуем шагам мастера установки и завершаем ее (Рисунок 3).

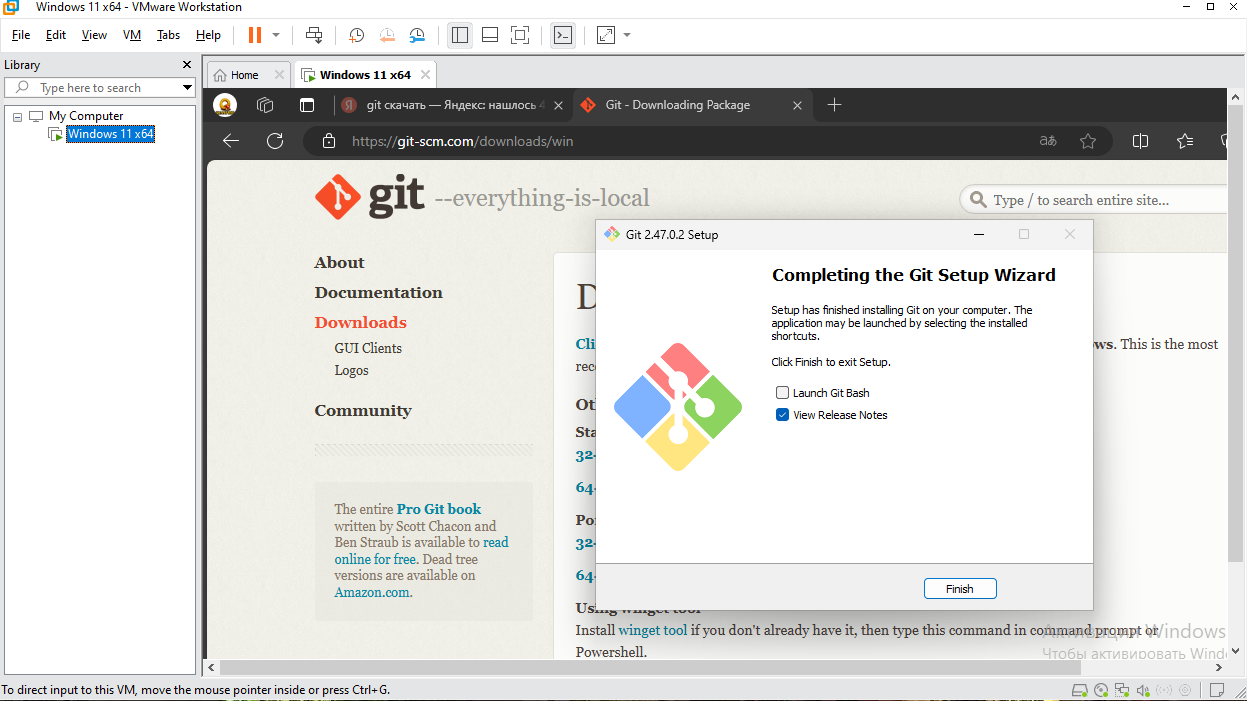


Рисунок 3 – Установка Git

Теперь создадим новую ветку от «main» ветки в нашем репозитории. Дадим ей название «LR4-TVA» (Рисунок 4).

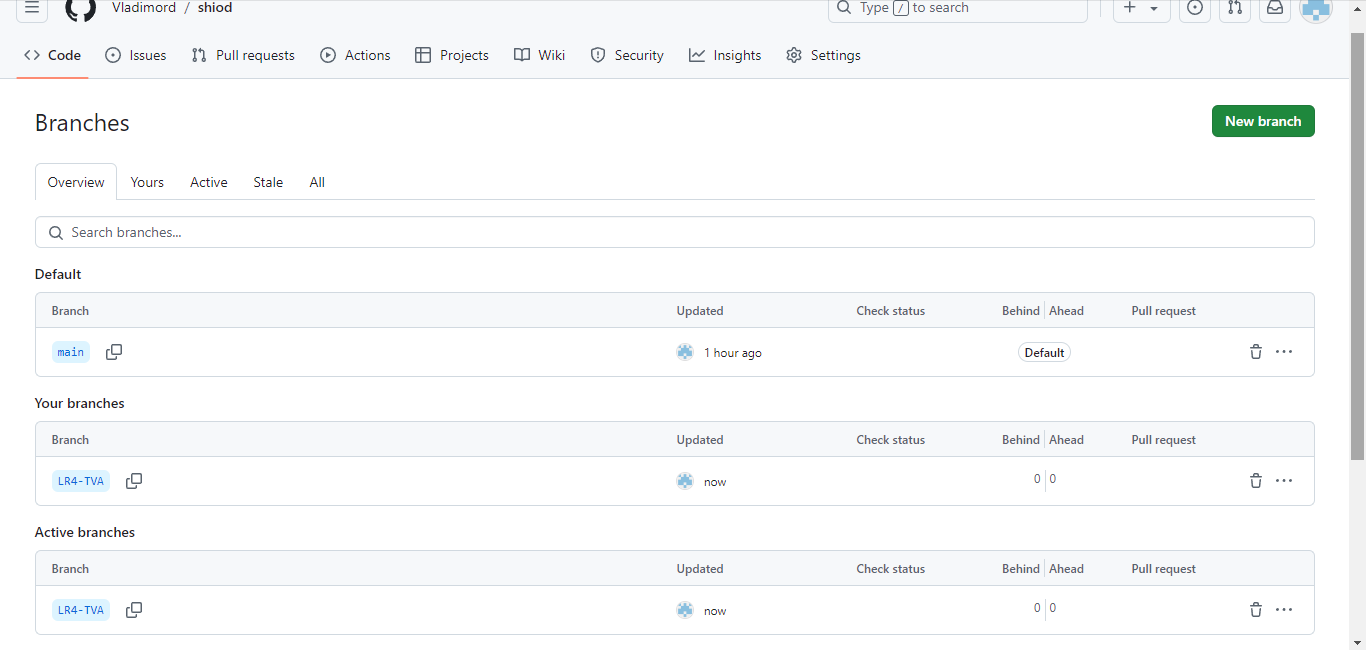


Рисунок 4 – Создание новой ветки от главной ветки «main»

Далее основные команды git: клонирование репозитория, инициализация, подготовка файлов к отправке, добавление коммита с комментарием, отправка изменений на удаленный репозиторий, слияние, pull request и др. напишем в файле «readme.md» в созданной ветке (Рисунок 5).

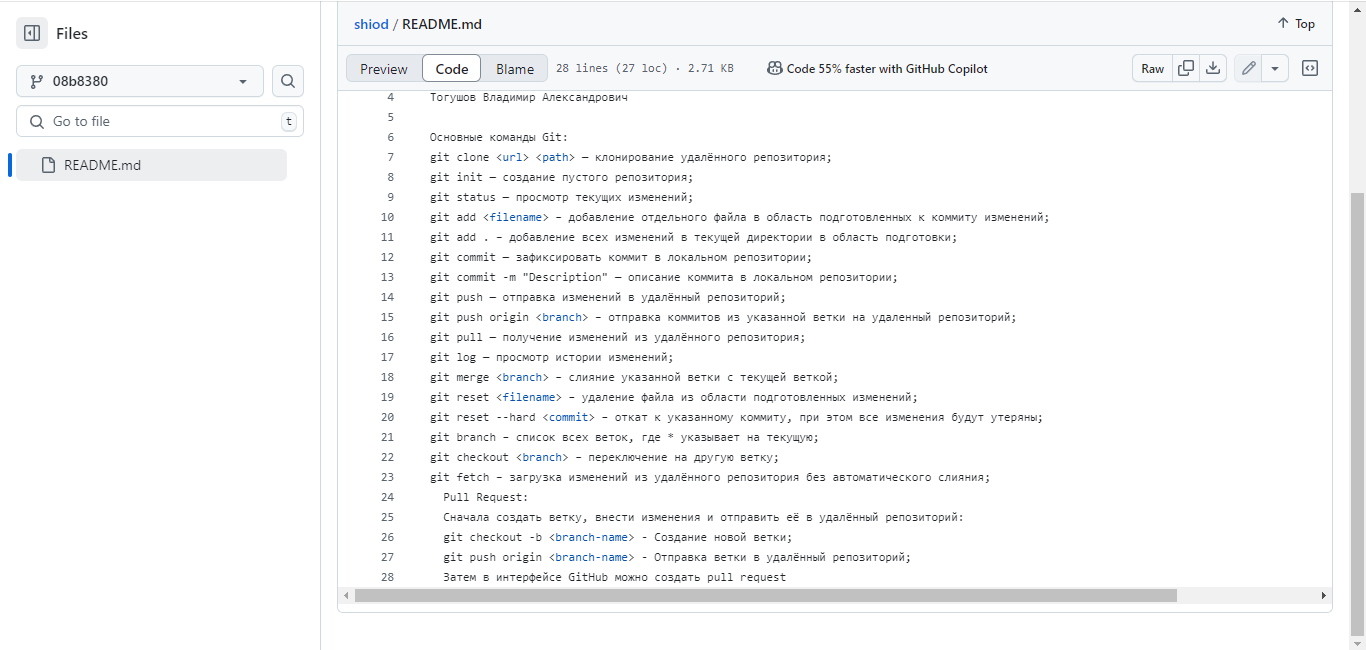


Рисунок 5 – Внесение основных команд Git в файл «readme.md»

Теперь можем клонировать удаленный репозиторий на локальный компьютер, где и будем вносить дальнейшие изменения уже в локальном репозитории.

Для этого запустим Git Bash. Выполним подключение к GitHub с помощью SSH-ключа.

Для генерации новой пары ключей в консоли git bash введем следующую команду:

|  |
| --- |
| ssh-keygen -t rsa -b 4096 -C "your\_email@example.com" |

Теперь нам нужно добавить сгенерированные ключи к ssh-агенту: eval $(ssh-agent -s).

Теперь нам нужно добавить созданные ключи к ssh-агенту. Сделаем мы это командой:

|  |
| --- |
| $ ssh-add ~/.ssh/id\_rsa. |

Добавляем публичный ключ на GitHub. Для этого открываем файл .pub с публичным ключом по пути, который мы оставили по умолчанию. Копируем содержимое файла в поле «Key» в разделе «settings/SSH and GPG keys». В поле «Title» вводим название ключа.

Выберем место для сохранения репозитория через «cd» и введем команду git clone <url> со ссылкой на наш репозиторий. После наш репозиторий появится в указанной папке, например Документы (Рисунок 6).

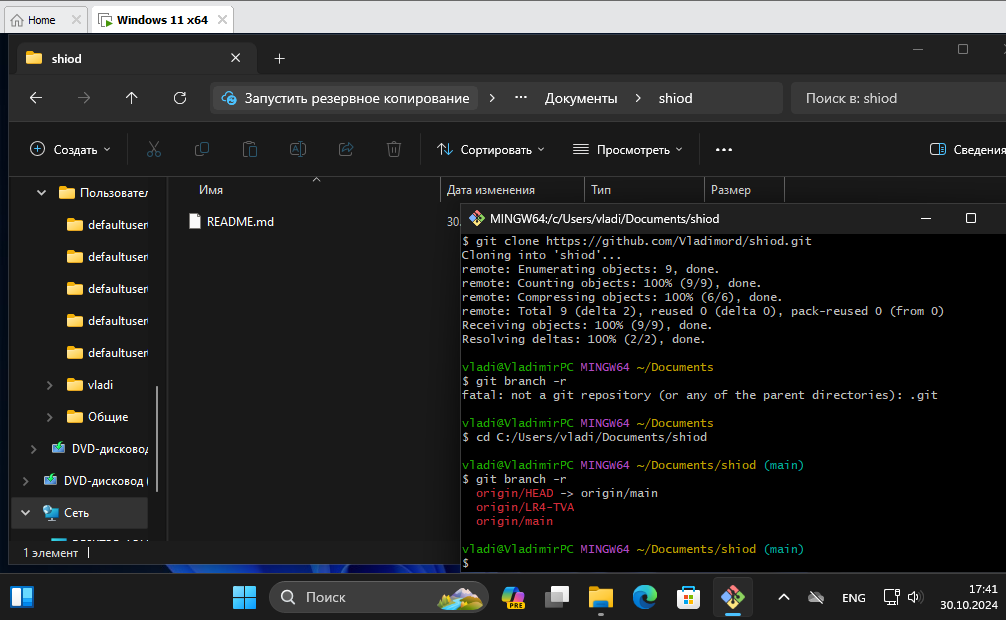


Рисунок 6 – Клонирование удаленного репозитория с GitHub

Теперь создадим в локальном репозитории текстовый файл с названием «FIO.txt» с данными о себе: ФИО, операционная система, на которой мы работаем с Git, специальность, место работы или направление будущей работы, ключевые навыки, увлечения, хобби (Рисунок 7).

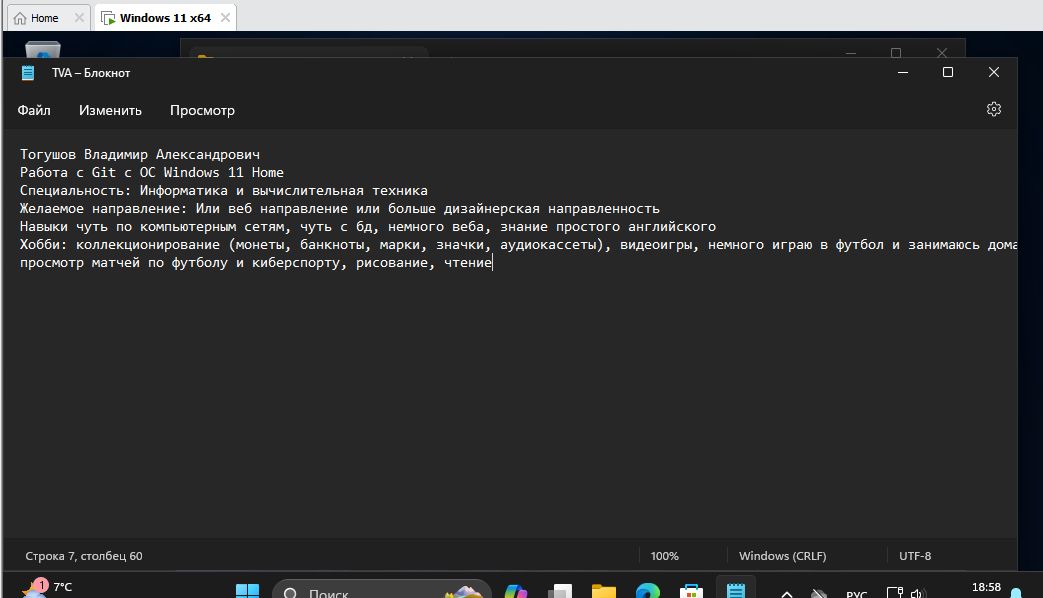


Рисунок 7 – Создание текстового файла с данными о себе

Затем сохраним изменения, будем работать в своей ветке, и отправим изменения из локального репозитория на удаленный.

Сначала переключимся на нашу ветку: git checkout <branch>.

Добавим все изменения в текущей директории в область подготовки: git add .

Сделаем коммит в Git с комментарием: git commit -m "Созданный локальный репозиторий связали с удаленным, переданы из локального в удаленный репозиторий: данные о себе".

Теперь настроим связь с удаленным репозиторием с помощью команды: git remote add <name> <url>.

Назначим в качестве ветки для синхронизации нужную нам ветку «LR4-TVA» с помощью команды: git push --set-upstream origin LR4-TVA.

Проверим, что изменения появились в репозитории на GitHub (Рисунок 8).

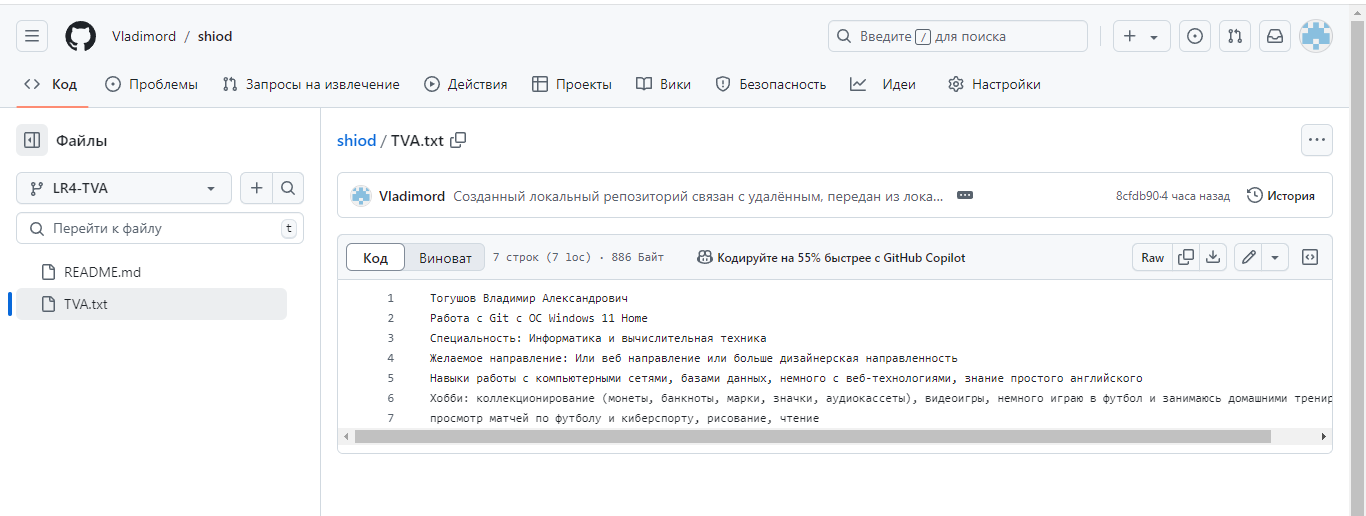


Рисунок 8 – Добавление текстового файла в удаленный репозиторий

Откроем в локальном репозитории созданный ранее файл «FIO.txt». Добавим данные о среднем балле по предыдущему образованию. Отправим изменения на удаленный репозиторий. Также используем команды: git add ./commit/push. Посмотрим на изменения на GitHub (Рисунок 9).

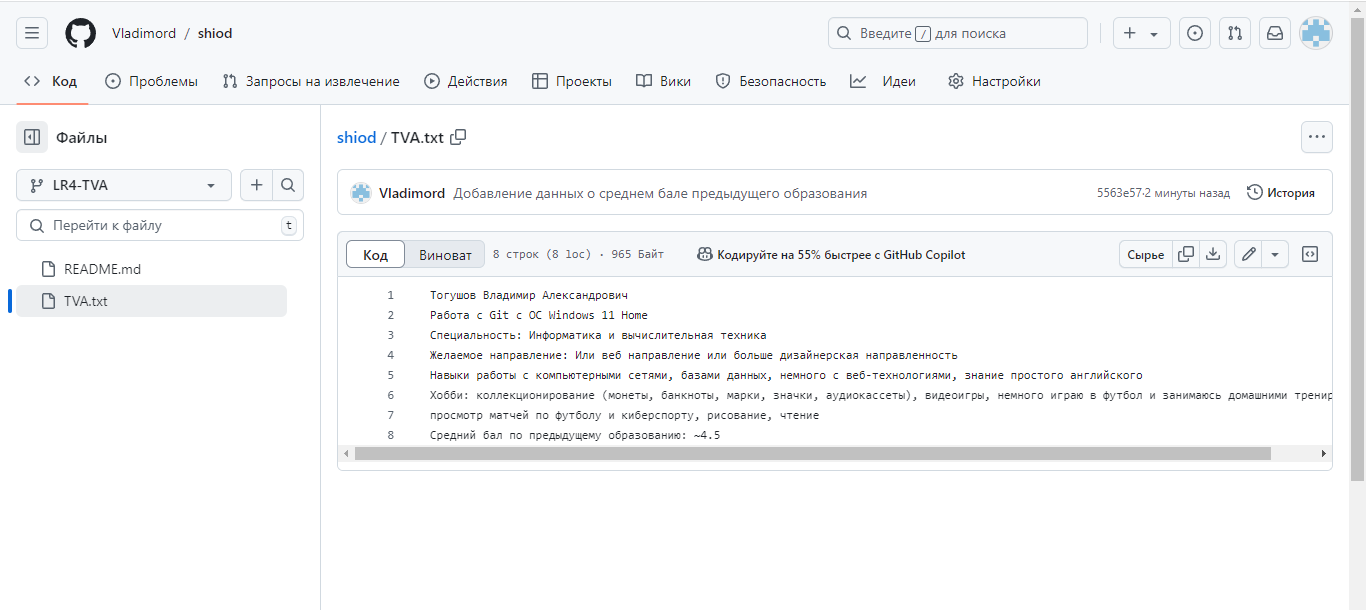


Рисунок 9 – Изменения файла с добавлением данных о среднем бале

Снова откроем наш текстовый файл на локальном репозитории и внесем данные о месте рождения. Также используем команды: git add ./commit/push. Посмотрим на изменения в удаленном репозитории (Рисунок 10).

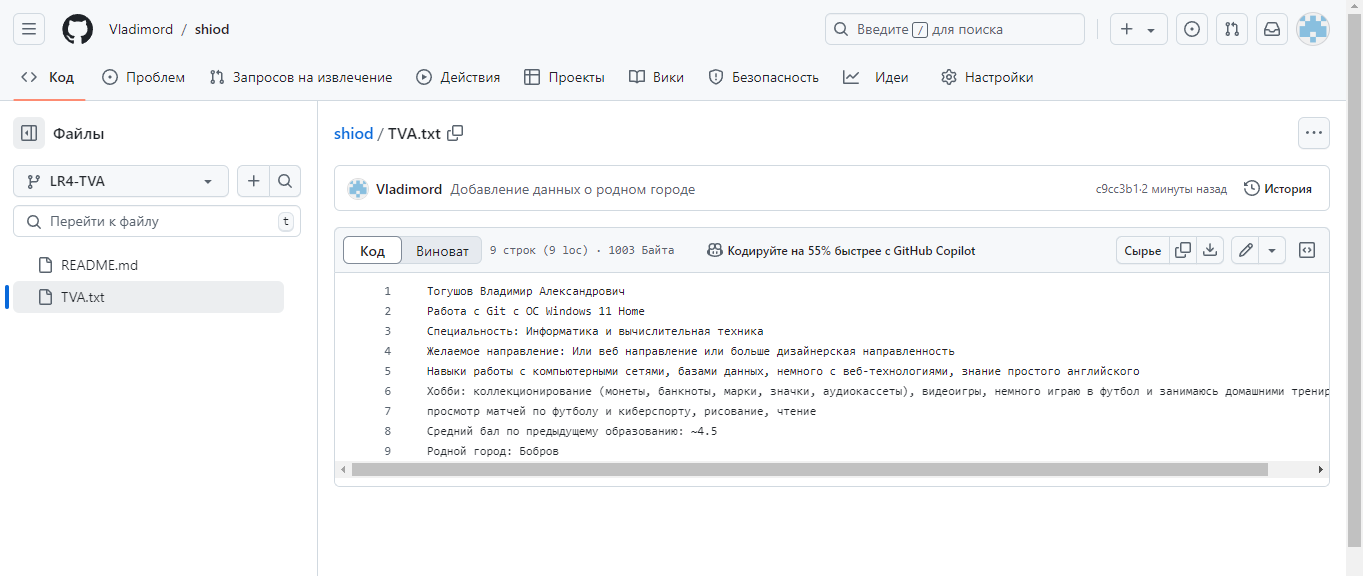


Рисунок 10 – Изменение файла с добавление данных о родном городе

Произведем откат изменений до предыдущего коммита. Итогом будет состояние нашего репозитория в нашей ветке до внесения информации о родном городе.

Для отката к определенному коммиту используем команду: git revert <commit id>, где commit id – id коммита, до которого нужно произвести откат. Либо другая команда: git revert HEAD. Проверим эти изменения на GitHub (Рисунок 11).

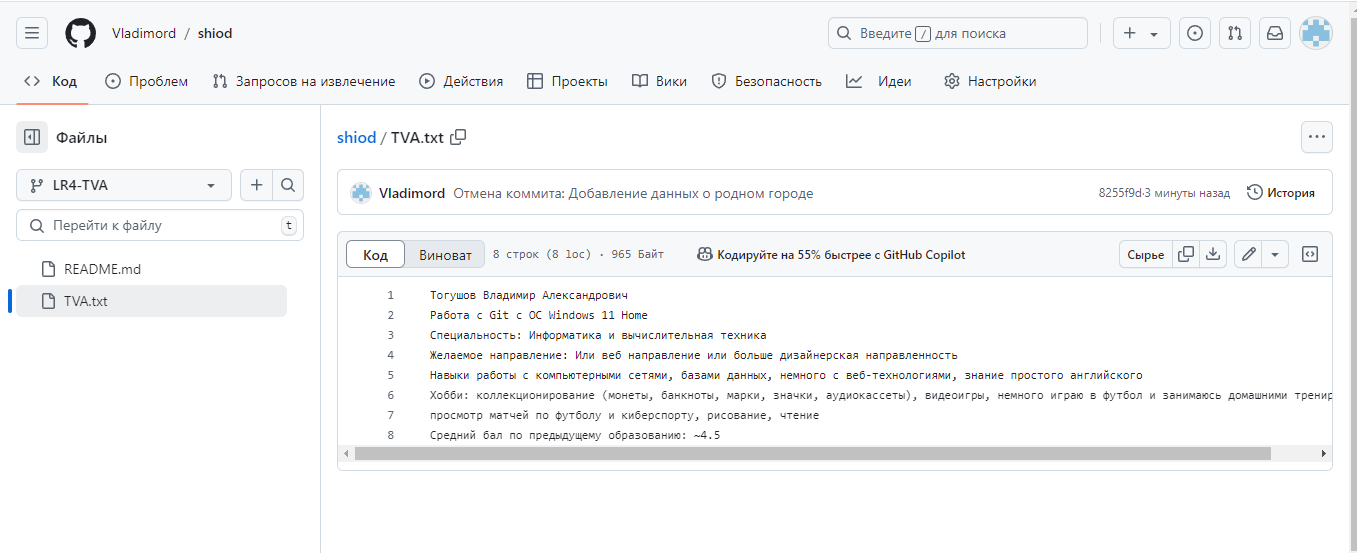


Рисунок 11 – Отмена последнего коммита с внесенными изменениями

Ссылка на репозиторий: <https://github.com/Vladimord/shiod.git>

Контрольные вопросы:

1) Что такое Git?

Git — это система контроля версий, которая помогает отслеживать историю изменений в файлах. Её используют программисты для совместной работы над проектами.

Git хранит не список изменений в файлах, а набор снимков — полное отображение того, как выглядит файл в момент сохранения. Это позволяет всегда иметь полную информацию обо всех файлах и быстро восстанавливать любую из предыдущих версий.

2) Для чего команда разработки использует Git?

Команда разработки использует Git для управления версиями исходного кода программ. Вот некоторые преимущества использования этой системы:

Прозрачная история изменений. Участники команды могут просматривать её с указанием авторов и вклада в разработку проекта.

Параллельная работа. Разработчикам не нужно ждать, пока другой допишет код, — можно работать параллельно.

Независимое сохранение проделанной работы. При работе над одним продуктом разработчики могут сохранять собственные изменения независимо друг от друга.

Возможность вернуться к предыдущей версии кода. Если в одной из веток разработка пойдёт не по плану и произойдёт ошибка, всё можно просто откатить до предыдущей ветки в системе контроля версий Git, где ошибок не было.

3) Основные команды и операции, необходимые для работы в Git.

* git init. Создание нового репозитория Git.
* git add. Добавление изменённых файлов в индекс, который позже будет использоваться для создания коммита.
* git commit. Создание нового коммита на основе текущего состояния индекса.
* git push. Отправка изменений из локального репозитория на сервер.
* git pull. Извлечение изменений с сервера и объединение их с локальным репозиторием.
* git branch. Создание новой ветки.
* git checkout. Переключение между ветками.
* git merge. Слияние изменений из одной ветки в другую.
* git status. Проверка статуса репозитория.
* git log. Просмотр журнала коммитов.

4) Что значит инициализация репозитория, какая команда для этого используется?

Инициализация репозитория в Git означает создание скрытой папки, в которой содержатся все объекты и ссылки, которые Git использует и создаёт в истории работы над проектом.

Для инициализации репозитория используется команда git init. Её выполняют только один раз для первоначальной настройки нового репозитория.