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

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

По дисциплине: «Системы хранения и обработки данных»

Тема: «Основы работы с системой управления версиями Git»

Выполнил работу студент группы мИИВТ-231 Черноусов М.К

Принял: Короленко В.В.

Защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Воронеж 2023

**Цель 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.

**Ход работы**:

В начале был создан профиль на платформе GitHub. (Рисунок 1).

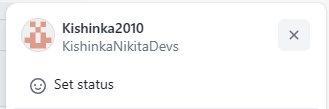


Рисунок 1 – Профиль на платформе github

Далее, для организации наших проектов, мы создали новый репозиторий. В процессе создания репозитория, указали его название, добавил краткое описание, и выбрал опции настройки, такие как публичный доступ, наличие README-файла. В README-файл была добавлена информация о данной лабораторной работе. Результат представлен на рисунке 2. Ссылка на репозиторий: https://github.com/Kishinka2010/LR4.git

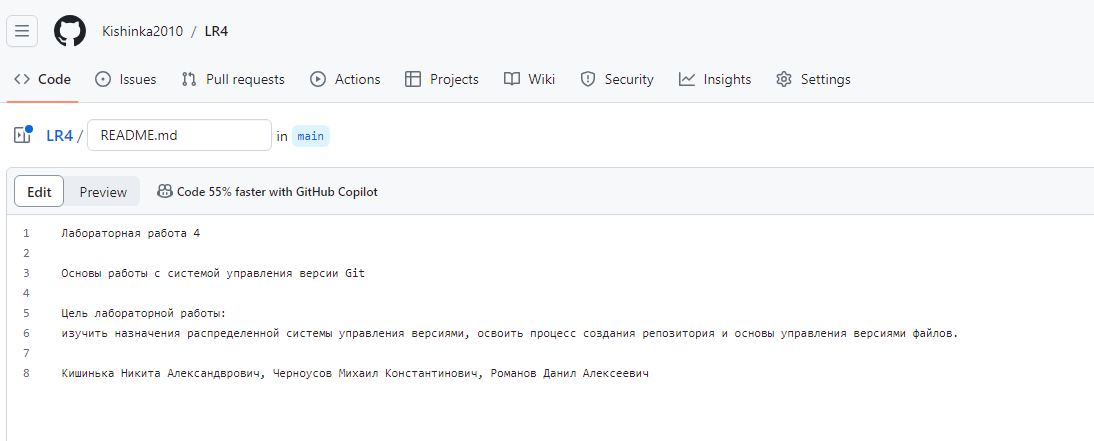


Рисунок 2 – Файл readme.md

Для отработки и внесения изменений в проект, мы создали новые ветки, которые назвали «LR4-KNA», «LR4-CHMK, «LR4-RDA». В этих ветках мы планируем проводить основную работу, внесение изменений и тестирование, сохраняя "main" в стабильном состоянии. Далее в этих ветках мы описали основные команды git. Результат работы представлен на рисунке 3.

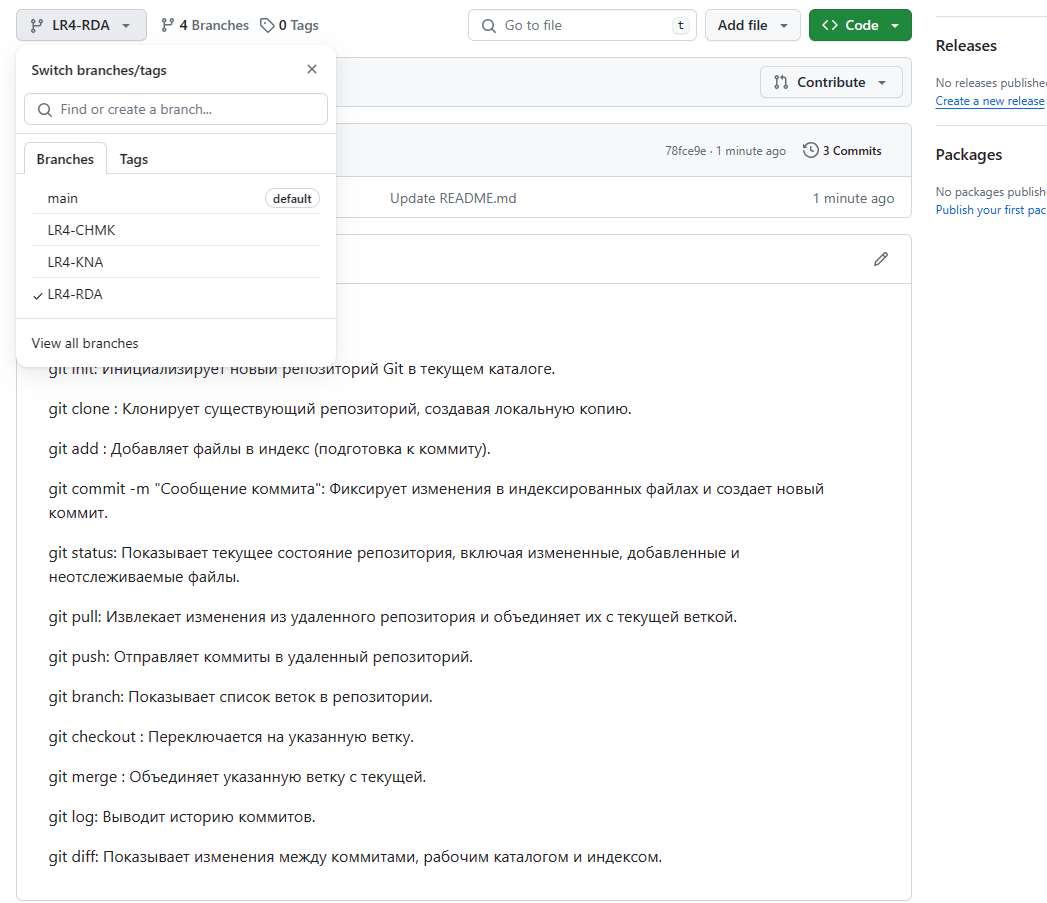


Рисунок 3 – Основные команды для git

Далее был клонирован удаленный репозиторий на локальный компьютер.

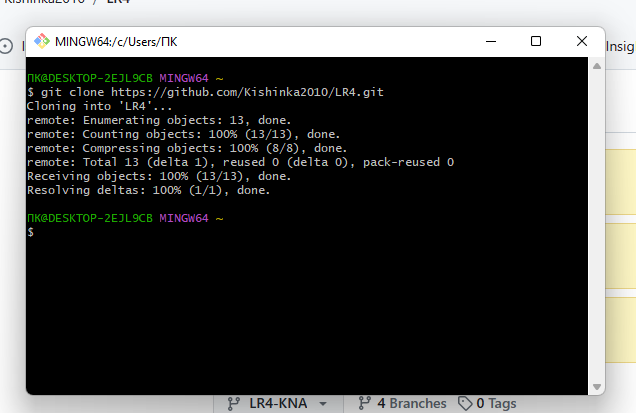


Рисунок 4 – Клонирование репозитория

После создадим файл “KNA.txt”, “CHMK.txt”, “RDA.txt” в локальных репозиторий с помощью команды: git add. После все изменения были отправлены на удаленный репозиторий с помощью команд: git commit –m “Add txt file” и git push. Результат работы представлен на рисунке 5.

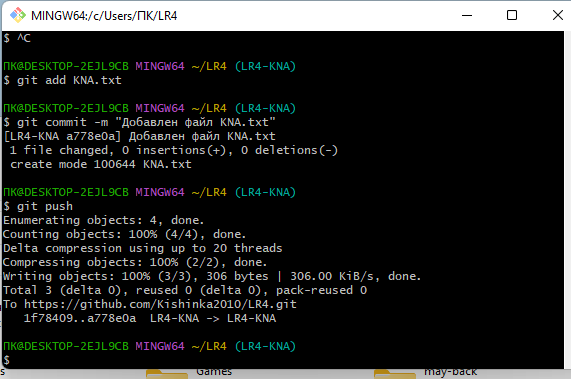


Рисунок 5 – Отправка изменений в ветку

Мы также можем посмотреть, какие файлы были добавлены в удаленных репозиториях. Эти изменения наглядно представлены на рисунках 6, 7 и 8.

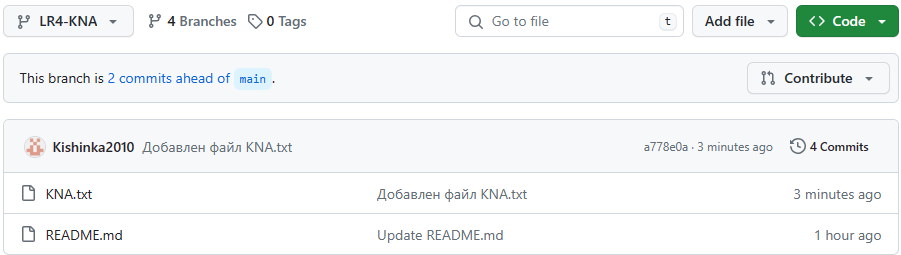


Рисунок 6 – Наличие изменений в ветке LR4-KNA

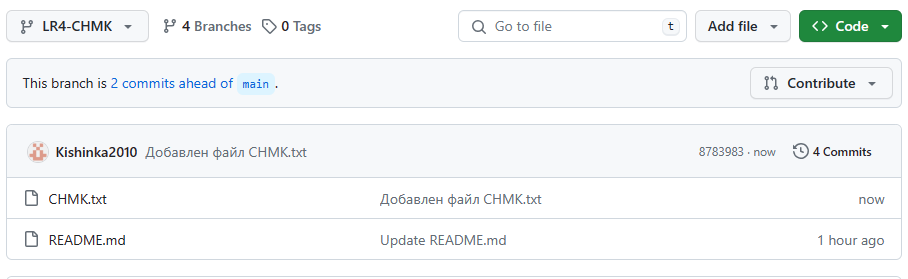


Рисунок 7 – Наличие изменений в ветке LR4-CHMK

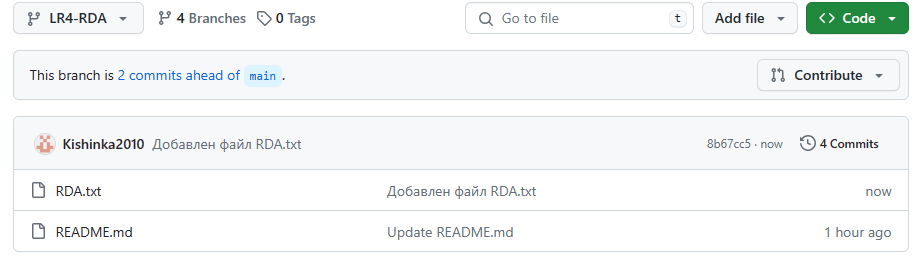


Рисунок 8 – Наличие изменений в ветке LR4-RDA

Далее внесем в текстовые файлы информацию о среднем балле студента, а после отправим на удаленный репозиторий. Процесс отправки изменений представлен на рисунках 9, 10 и 11

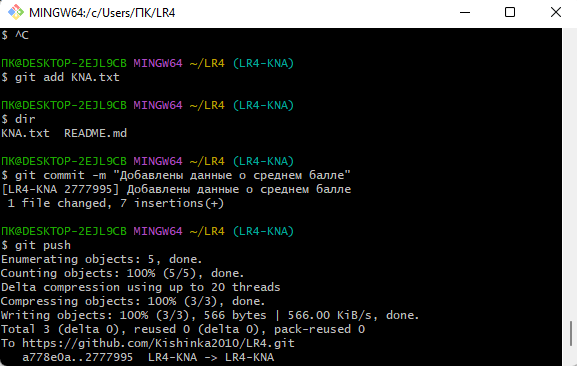


Рисунок 9 - Отправка изменений со средним баллом в ветку LR4-KNA

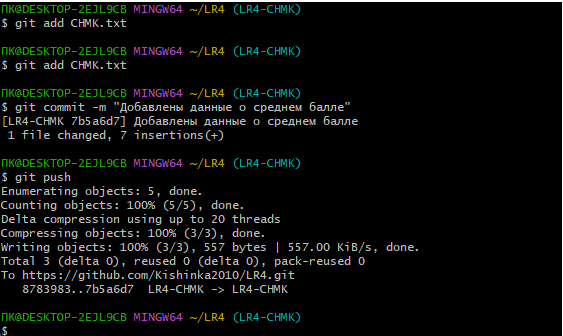


Рисунок 10 - Отправка изменений со средним баллом в ветку LR4-CHMK

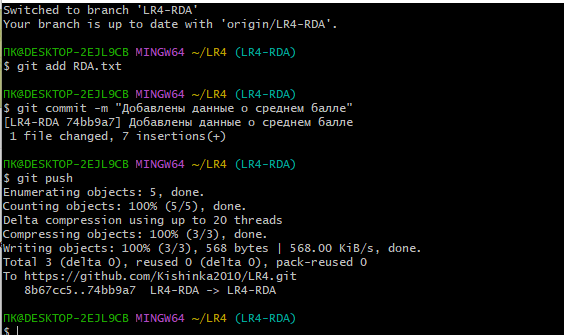


Рисунок 11 - Отправка изменений со средним баллом в ветку LR4-RDA

Мы можем увидеть на рисунке 12 изменения, представленные в ветке LR4-KNA, чтобы убедиться, что все изменения успешно загружены.

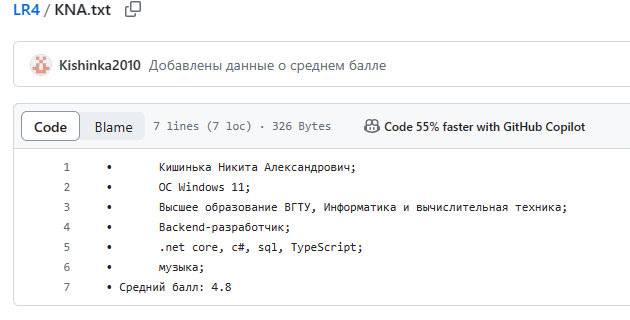


Рисунок 12 – Текстовый файл со средним баллом

Далее добавим информацию о месте рождения и отправим на удаленные репозитории. Процесс отправки изменений представлен на рисунках 13, 14, 15

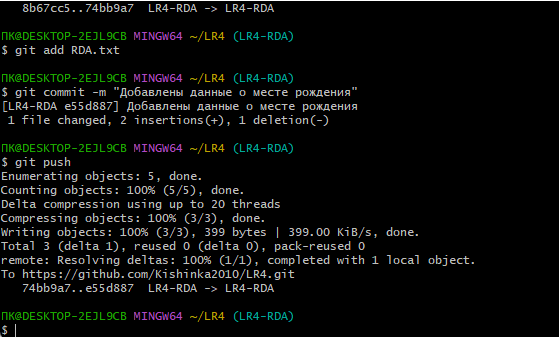


Рисунок 11 - Отправка изменений о месте рождения в ветку LR4-RDA

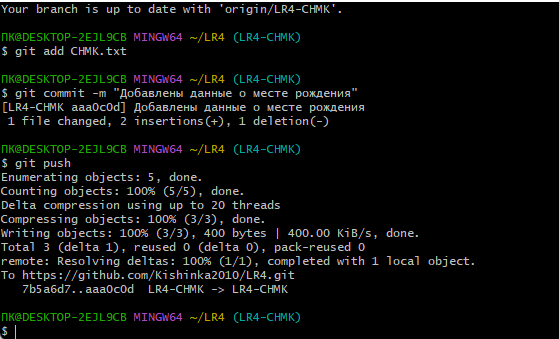


Рисунок 12 - Отправка изменений о месте рождения в ветку LR4-CHMK

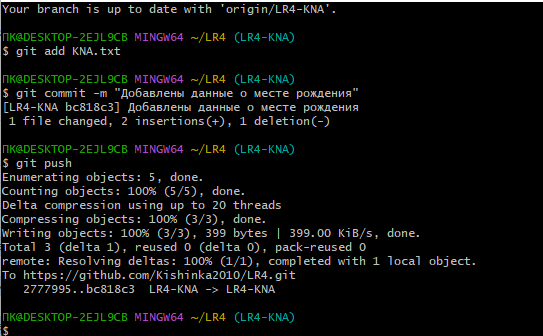


Рисунок 13 - Отправка изменений о месте рождения в ветку LR4-KNA

Также в файле мы можем увидеть добавленные изменения. Результат представлен на рисунке 14

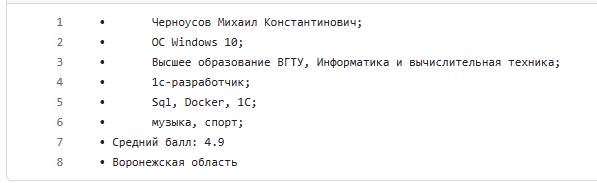


Рисунок 14 – Текстовый файл с местом рождения

После выполним откат изменений до предыдущего коммита при помощи команды "git reset --hard <Хеш>". Здесь "Хеш" представляет собой уникальный идентификатор (SHA-1 хеш) коммита, к которому мы хотим вернуться. Этот процесс позволит нам отменить все изменения после указанного коммита и вернуть репозиторий к состоянию, соответствующему выбранному коммиту.

На рисунке 15 мы можем видеть, что команда успешно выполнилась.

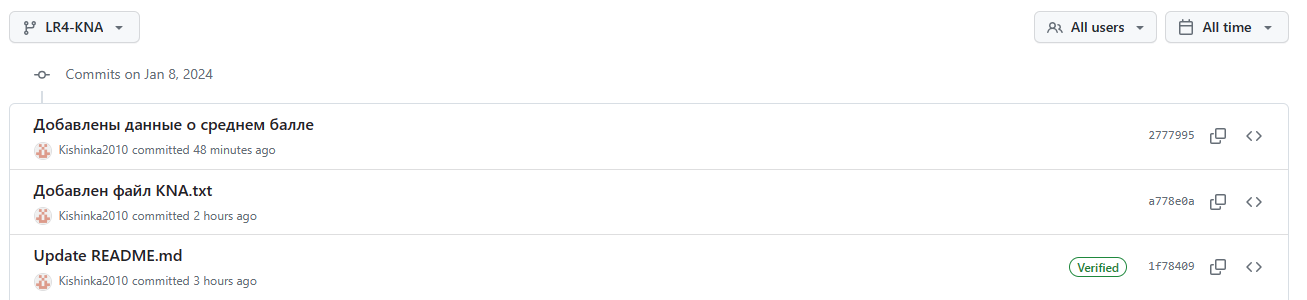


Рисунок 15 – Коммиты

Также в файле мы можем увидеть, что изменения откатились. Результат представлен на рисунке 16

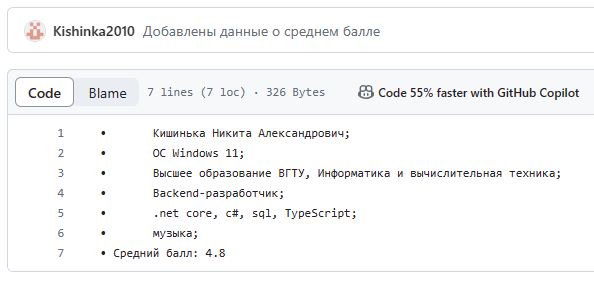


Рисунок 16 – Файл после отката

**Вывод**: в данной лабораторной работе было изучено назначение распределённой системы управления версиями, освоен процесс создания репозитория и основы управления версиями файлов.

Контрольные вопросы

⦁ Что такое Git?

Git — это система управления версиями, предназначенная для отслеживания изменений в исходном коде программного обеспечения и координации работы нескольких программистов над одним проектом. Она позволяет отслеживать изменения в коде, возвращаться к предыдущим версиям, объединять изменения от разных разработчиков и многое другое.

⦁ Для чего команда разработки использует Git?

Команда разработки использует Git для эффективного управления версиями и координации работы над программным обеспечением. Git предоставляет средства отслеживания изменений в коде, позволяет создавать ветви для параллельной разработки, а также управлять и объединять изменения от различных участников проекта. Это обеспечивает надежное и структурированное сотрудничество в команде, позволяет возвращаться к предыдущим версиям кода, а также обеспечивает безопасное внесение и тестирование изменений перед интеграцией в общий кодовую базу. Git также предоставляет возможности работы с удаленными репозиториями, что позволяет разработчикам с легкостью обмениваться своей работой и поддерживать ее в синхронизированном состоянии.

⦁ Основные команды и операции, необходимые для работы в Git.

⦁ Основные команды и операции, необходимые для работы в Git.

⦁ git init - инициализация нового репозитория.

⦁ git clone - клонирование удаленного репозитория на локальную машину.

⦁ git add - добавление изменений в текущей директории в индекс.

⦁ git commit - фиксация изменений в индексе.

⦁ git push - отправка локальных изменений в удаленный репозиторий.

⦁ git pull - получение и объединение изменений из удаленного репозитория в локальный репозиторий.

⦁ git branch - создание, удаление и список веток.

⦁ git merge - объединение другой ветки с текущей веткой.

⦁ git checkout - переключение между ветками и восстановление файлов из репозитория.

⦁ git status - отображение состояния файлов в рабочем каталоге и индексе.

⦁ Что значит инициализация репозитория, какая команда для этого используется?

Инициализация репозитория в Git означает создание нового репозитория или превращение существующего каталога в Git-репозиторий. Когда вы инициализируете репозиторий, Git создает необходимую структуру данных и файлы для отслеживания изменений в вашем проекте. Это позволяет вам использовать Git для управления версиями кода. Для инициализации нового репозитория в Git используется команда git init. Выполнение этой команды внутри каталога проекта создает скрытую подкаталог .git, который содержит все необходимые файлы и данные для работы с Git.