МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

«Исследование возможностей Git для работы с локальными репозиториями»

Отчет по лабораторной работе № 1.1

по дисциплине «Основы программной инженерии»

Выполнил студент группы ПИЖ-б-о-21-1

Трушева В. О. .«20» сентября 2022г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверила Воронкин Р.А. _____
(подпись)

Ставрополь 2022

Цель работы: исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

1. Ответы на контрольные вопросы

1. Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

2. В чем недостатки локальных и централизованных СКВ?

Недостатки локальных СКВ: возможность потери данных вследствие возникновения физических поломок оборудования, отсутствие возможности совместной разработки.

Недостатки централизованных СКВ при временном выключение сервера останавливает работу программистов, они не могут ни сохранять новые варианты версий, ни взаимодействовать между собой. В случае же повреждения диска, на котором хранится центральная база данных, все наработки по проекту теряются безвозвратно.

3. К какой СКВ относится Git?

К распределённым системам контроля версий (РСКВ). В РСКВ клиенты не просто скачивают снимок всех файлов (состояние файлов на определённый момент времени) – они полностью копируют репозиторий.

4. В чем концептуальное отличие Git от других СКВ?

Основное отличие Git от любой другой СКВ – это подход к работе со своими данными. Концептуально, большинство других систем хранят

информацию в виде списка изменений в файлах. Эти системы (CVS, Subversion, Perforce, Bazaar и т. д.) представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени (обычно это называют контролем версий, основанным на различиях). Git не хранит и не обрабатывает данные таким способом.

5. Как обеспечивается целостность хранимых данных в Git?

Перед сохранением любого файла Git вычисляет контрольную сумму, и она становится индексом этого файла. Поэтому невозможно изменить содержимое файла или каталога так, чтобы Git не узнал об этом.

6. В каких состояниях могут находится файлы в Git? Как связаны эти состояния?

У Git есть три основных состояния, в которых могут находиться файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged).

Изменённый означает, что вы изменили файл, но ещё не зафиксировали его в своем локальном репозитории.

Индексированный – это изменённый файл, текущую версию которого вы отметили для включения в следующий коммит (для фиксации в своём локальном репозитории).

Зафиксированный означает, что файл уже сохранён в вашем локальном репозитории.

7. Что такое профиль пользователя в GitHub?

Профиль пользователя - это публичная страница на GitHub, как и в социальных сетях, в которой могут храниться проекты пользователя, версии этих проектов, также можно дать доступ другим пользователям для

изменения ваших проектов, и можно предлагать свои изменения другим пользователям.

8. Какие бывают репозитории в GitHub?

Удаленный репозиторий – это версии вашего проекта, сохраненные на удаленном сервере.

Локальный репозиторий – это репозиторий, которой хранится на компьютере, в рабочей папке проекта.

9. Укажите основные этапы модели работы с GitHub.

Создается репозиторий на GitHub. После этого необходимо клонировать его на ваш компьютер. Далее изменяете файлы в вашей рабочей копии. Выборочно вносите изменения, которые хотите включить в свой следующий коммит. И выполняете коммит, который включает ваши изменения.

10. Как осуществляется первоначальная настройка Git после установки?

Для начала нужно ввести свое имя пользователя и адрес электронной почты, также можно проверить версию Git, с помощью команды *git version.*

11. Опишите этапы создания репозитория в GitHub.

Для того чтобы создать репозиторий, необходимо в правом верхнем углу, рядом с аватаром есть кнопка с плюсиком, нажимая которую мы переходим к созданию нового репозитория. В результате будет выполнен переход на страницу создания репозитория. Наиболее важными на ней являются следующие поля:

- Имя репозитория. Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиториев, которые вы создавали.
- Описание (Description). Можно оставить пустым.
- Public/private. Выбираем открытый (Public), НЕ ставим галочку “Initialize this repository with a README” (В README потом будет лежать какая-то основная информация, что же такое ваш проект и как с ним работать).
- .gitignore и LICENSE можно сейчас не выбирать.

  После заполнения этих полей нажимаем кнопку Create repository.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Без лицензии применяются законы об авторском праве по умолчанию, что означает, что вы сохраняете все права на свой исходный код, и никто не может воспроизводить, распространять или создавать производные на основе вашей работы.

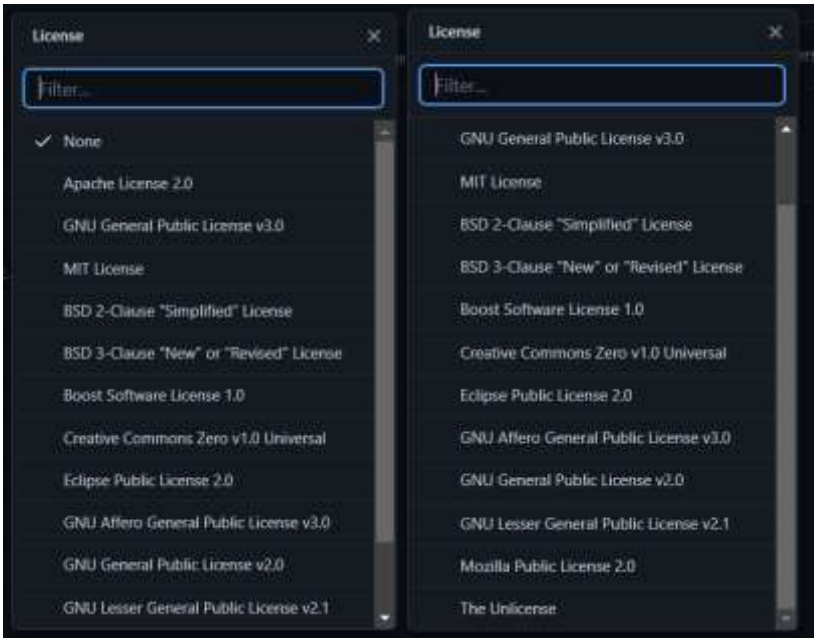

Рисунок 1.1 – Виды лицензий на GitHub

13. Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий?

Клонирование репозитория осуществляется с помощью данной команды: *git clone <ссылка на адрес репозитория>*.

Клонирование репозитория на компьютер нужно для того, чтобы локально хранить последние изменения вашего проекта, не затрагивая работу других пользователей при ветвлении.

14. Как проверить состояние локального репозитория Git?

Можно проверить с помощью команды *git status.*

15. Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/измененного файла под версионный контроль с помощью команды git add ; фиксации (коммита) изменений с помощью команды git commit и отправки изменений на сервер с помощью команды git push?

16. У Вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы можете осуществлять работу над некоторым проектом с использованием этого репозитория. Опишите последовательность команд, с помощью которых оба локальных репозитория, связанных с репозиторием GitHub будут находиться в синхронизированном состоянии. Примечание: описание необходимо начать с команды git clone .

Для начала необходимо скопировать проект с папкой себе на компьютер в определённое место с помощью команды: *git clone <адрес репозитория>*. И чтобы изменения, сделанные на компьютере сохранялись на удаленном репозитории, необходимо эти имзение добавить (git add), затем закоммитить (git commit –m “пояснение коммита”), а потом запушить (git push).

17. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.

18. Интерфейс командной строки является не единственным и далеко не самым удобным способом работы с Git. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git? Приведите как реализуются описанные в лабораторной работе операции Git с помощью одного из таких программных средств.

Github Desktop — программы под Windows 7+ и OS X, которая дублирует функциональность сайта github.com, но при этом работает локально на компьютере разработчика.

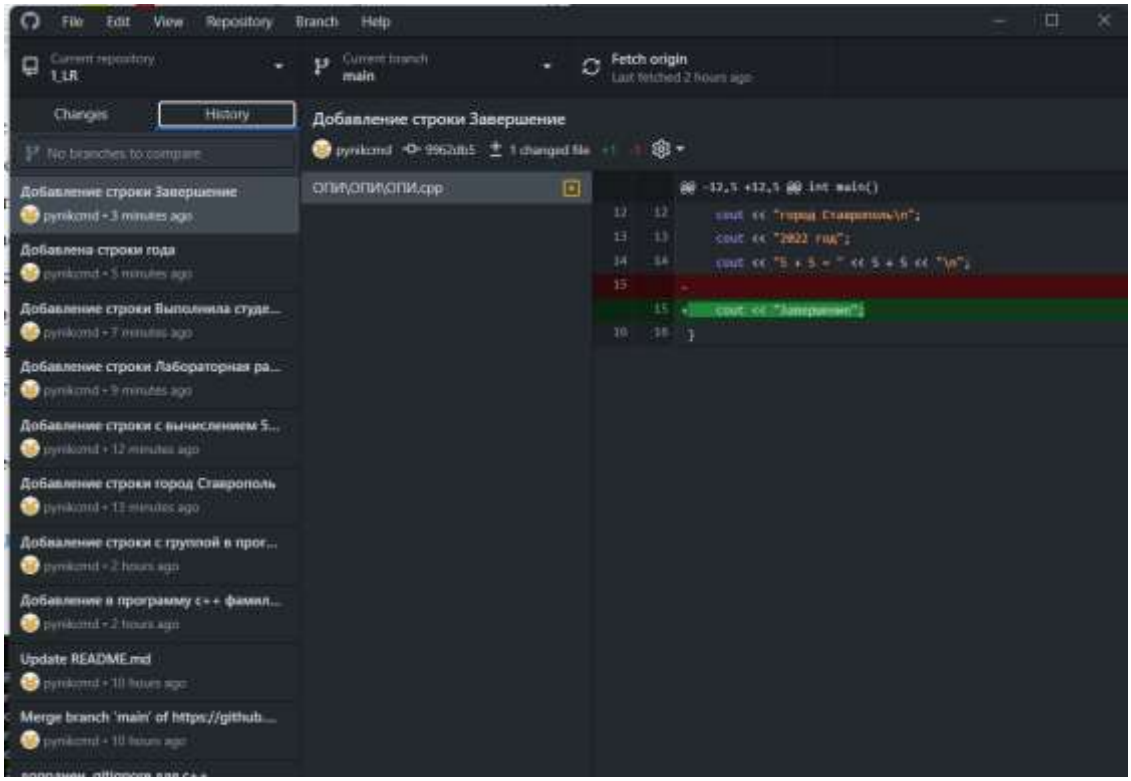

Рисунок 1.2 – Окно истории изменений

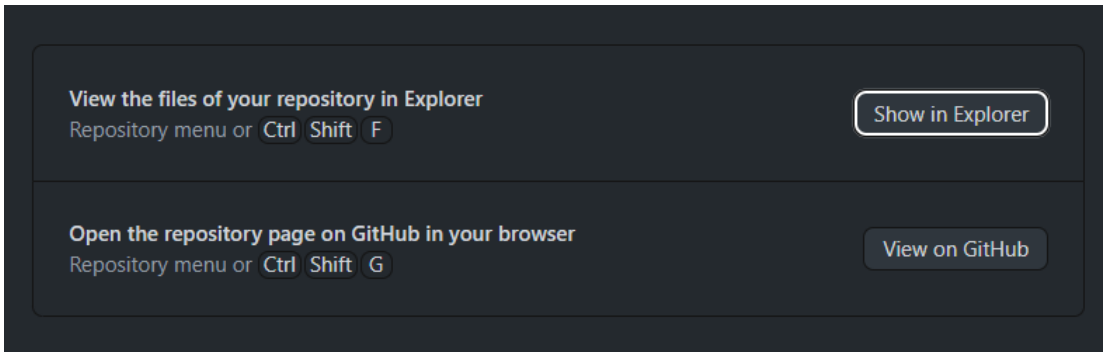

Рисунок 1.3 – Окно с ссылками на просмотр локальных файлов и репозитория на GitHub

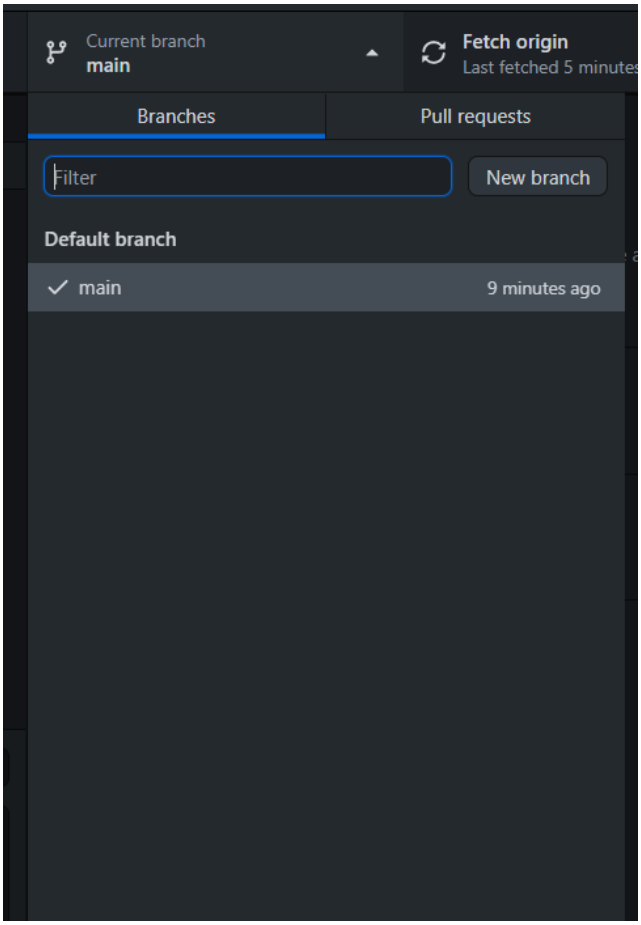

Рисунок 1.4 – Окно выбора текущей ветки

2. Практическая часть

Ссылка на git:

https://github.com/pynikcmd/1_LR

Клонирование репозитория на компьютер.

```
d:\fgit>git clone https://github.com/pynikcmd/1_LR.git
Cloning into '1_LR'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

d:\fgit>
```

Рисунок 2.1 – Результат клонирования репозитория

Проверка статуса.

```
d:\fgit\1_LR>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Рисунок 2.2 – Результат проверки статуса

Дополняем .gitignore необходимыми правилами для языка c++. И коммитим.

```
d:\fgit\1_LR>git add .

d:\fgit\1_LR>git commit
hint: Waiting for your editor to close the file... dos2uni
Aborting commit due to empty commit message.

d:\fgit\1_LR>git commit -m "дополнен .gitignore для c++"
[main c4645da] дополнен .gitignore для c++
 1 file changed, 351 insertions(+), 28 deletions(-)
```

Рисунок 2.3 – Результат коммита

Делаем push для сохранения проделанных действий в удаленный репозиторий.

```
d:\fgit\1_LR>git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 748 bytes | 187.00 KiB/s, done.
From https://github.com/pynikcmd/1_LR
   55a233e..681138e  main       -> origin/main
Merge made by the 'ort' strategy.
 README.md | 4 ++++
 1 file changed, 4 insertions(+)
 create mode 100644 README.md

d:\fgit\1_LR>git push
Enumerating objects: 9, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 3.21 KiB | 3.21 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/pynikcmd/1_LR.git
   681138e..b6e3ce8  main -> main
```

Рисунок 2.4 – Результат сохранения

Создаем проект в Visual Studio в папке с репозиторием и коммитим это. Проверяем git status.

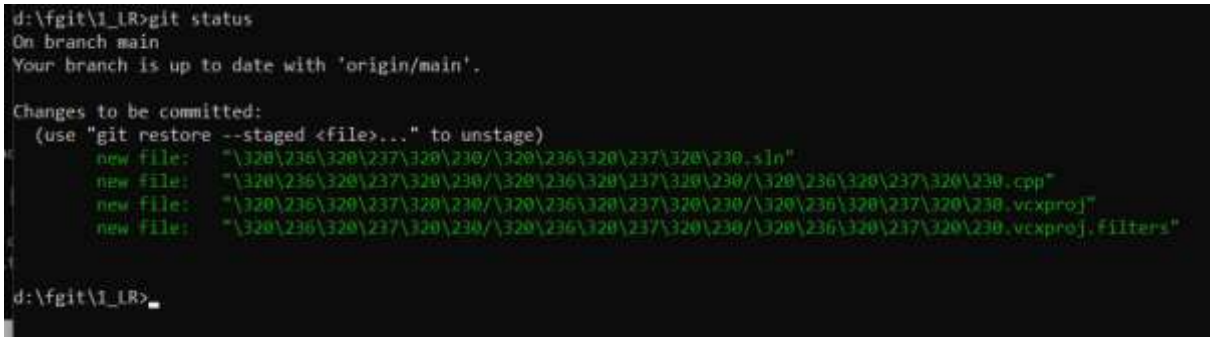

Рисунок 2.5 – Результат команды git status

Добавляя новые строчки, сохраняем проделанные действия с помощью команд git add и git commit. А также иногда вводим команду git push, чтобы наши изменения сохранялись в удаленном репозитории.

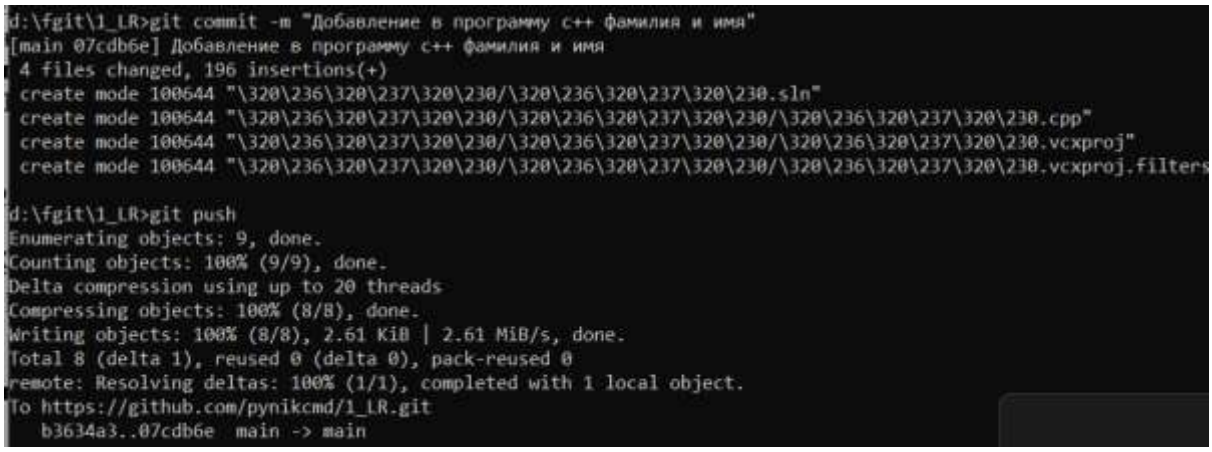

Рисунок 2.6 – Результат сохранения новой строки в коде

И с помощью команды *git log* можно проверить свои коммиты.

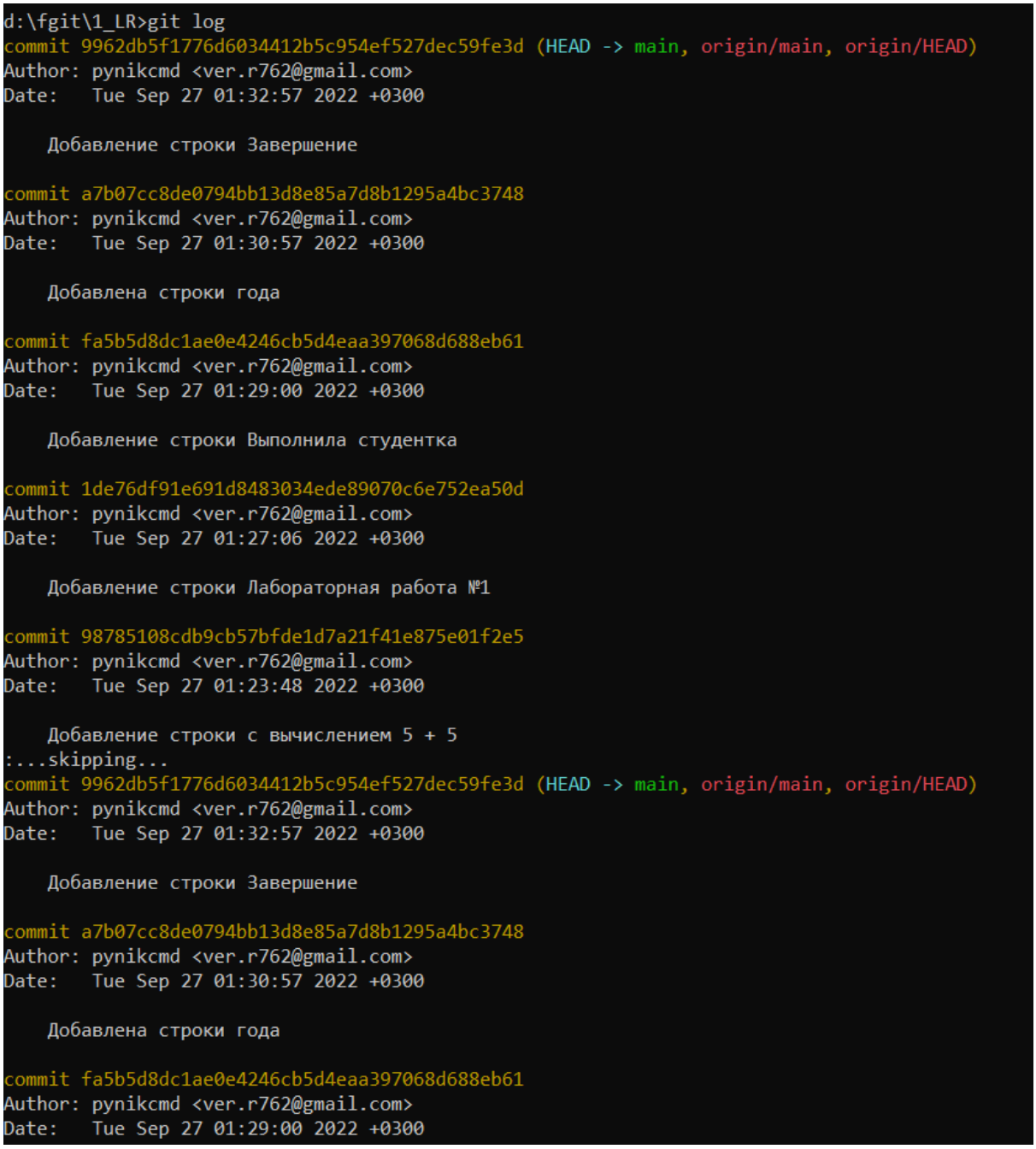

Рисунок 2.7 – Результат работы команды *git log*

Вывод: в ходе лабораторной работы были изучены базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.