

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

Кафедра инфокоммуникаций

**«Исследование основных возможностей Git и GitHub»
Отчет по лабораторной работе № 1.1
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-22-1
Душин Александр Владимирович.
Подпись студента _____
Работа защищена « » _____ 20__ г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

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

Ход работы:

1. Создадим общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный язык программирования:

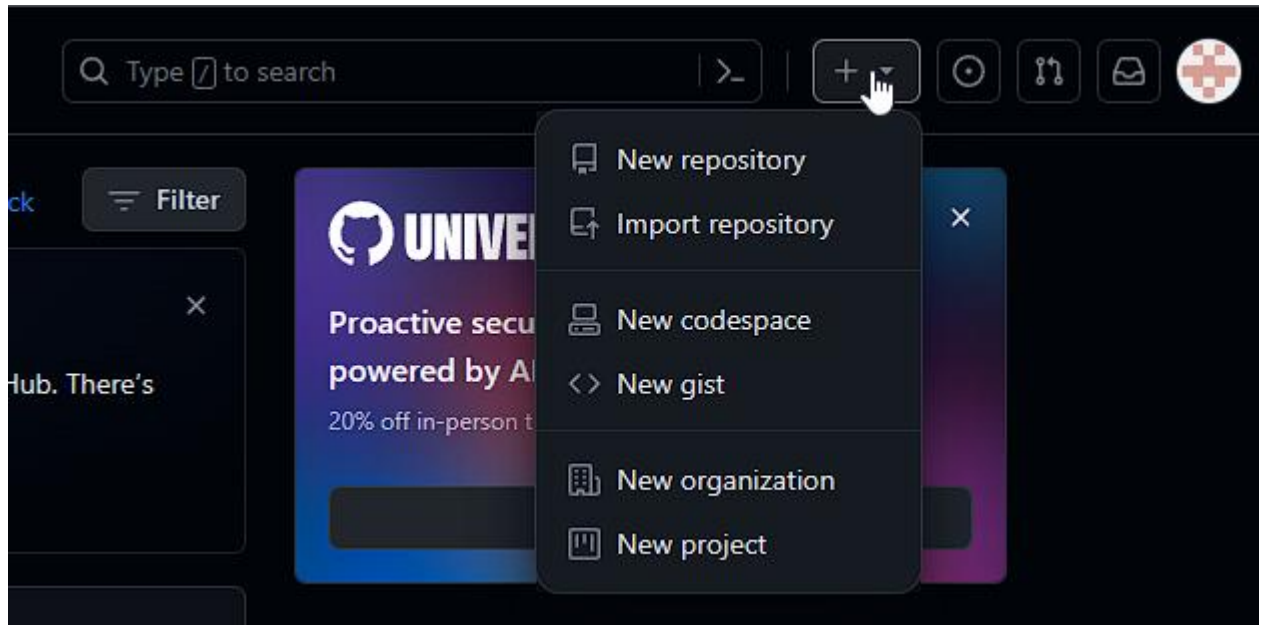


Рисунок 1 – Создание нового репозитория

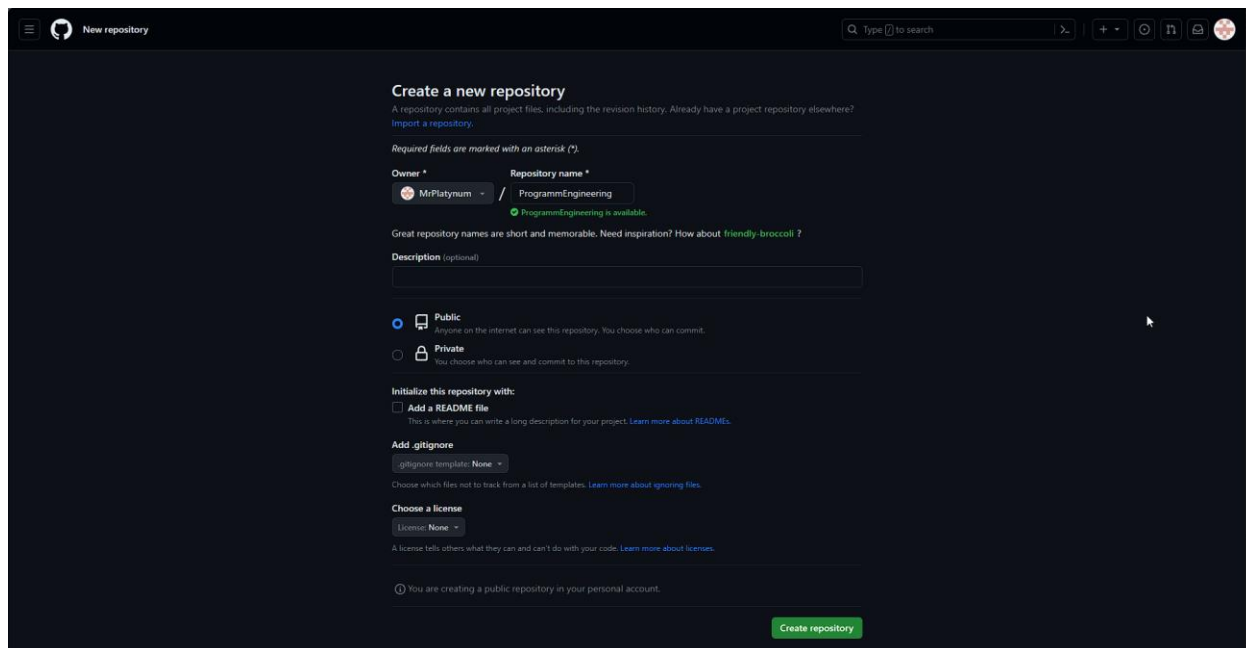


Рисунок 2 – Страница создания нового репозитория

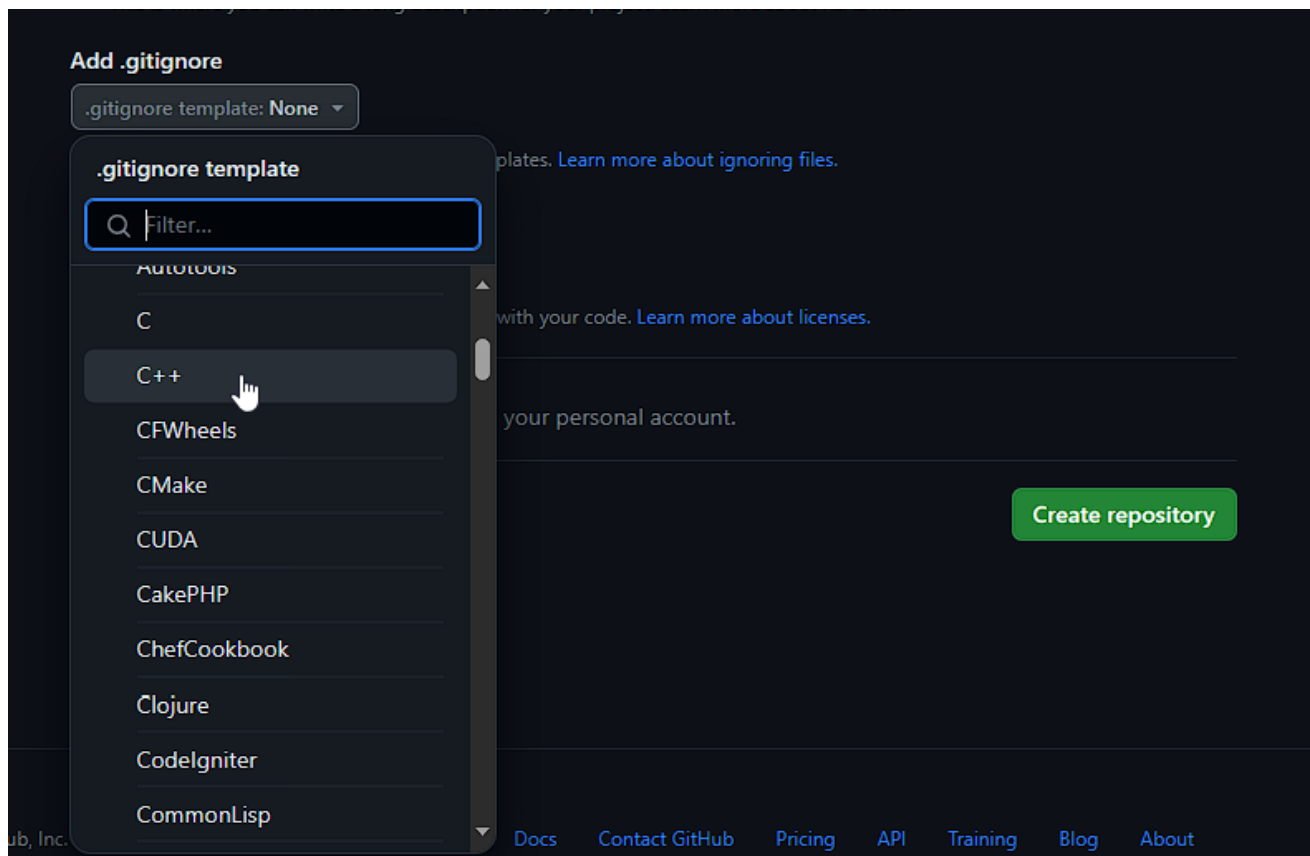


Рисунок 3 – Выбор языка программирования

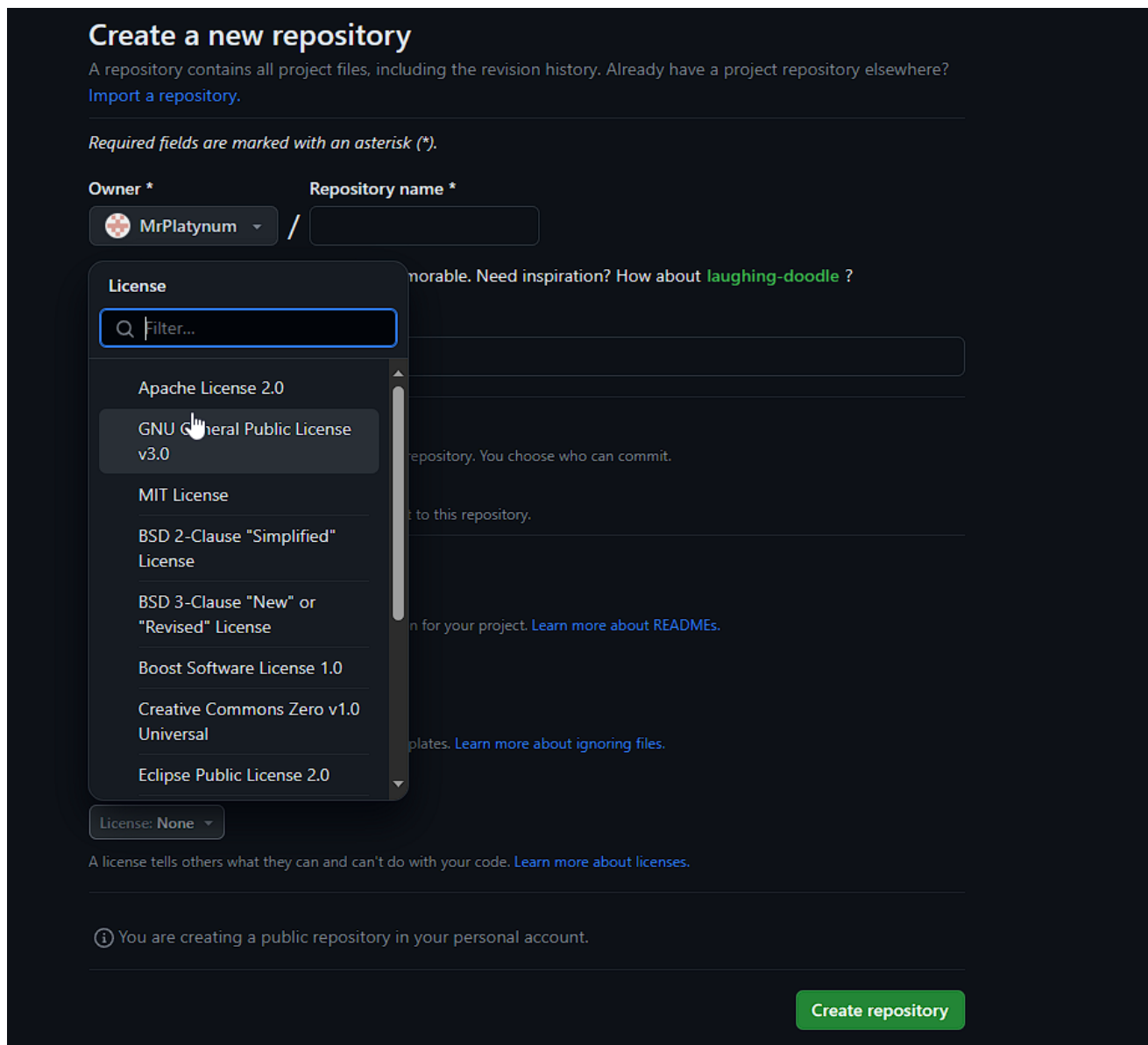


Рисунок 4 – Выбор лицензии

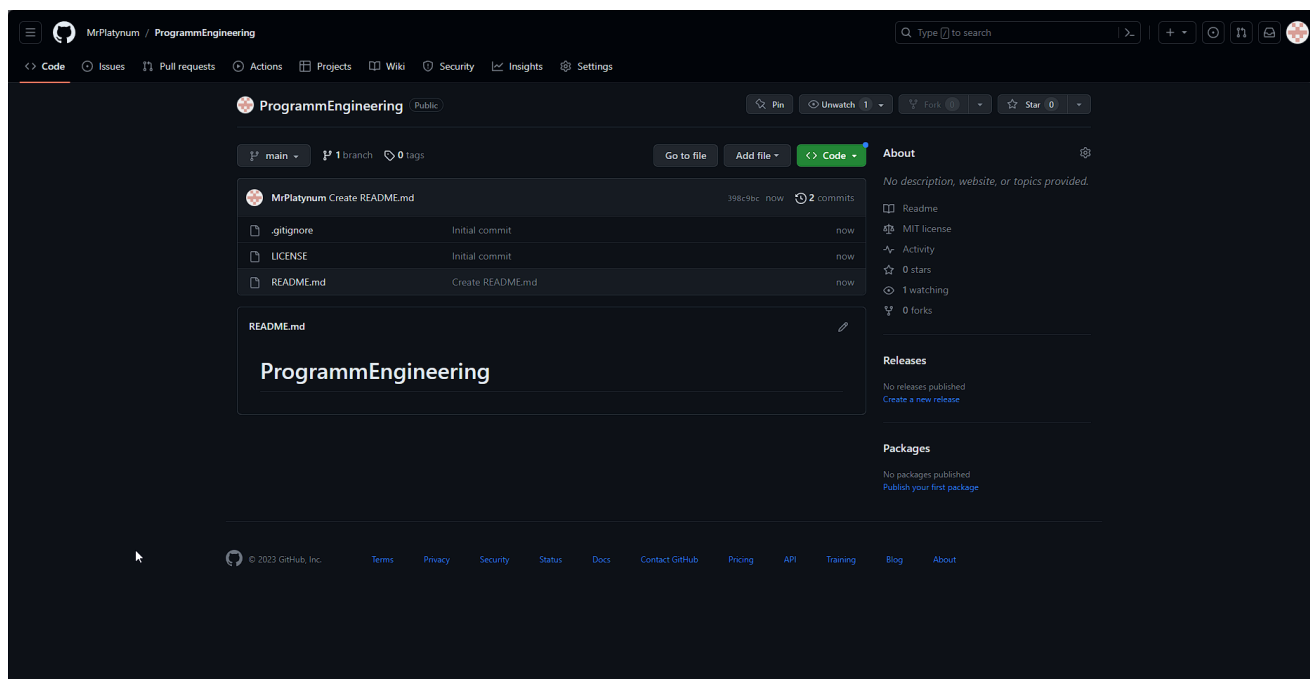
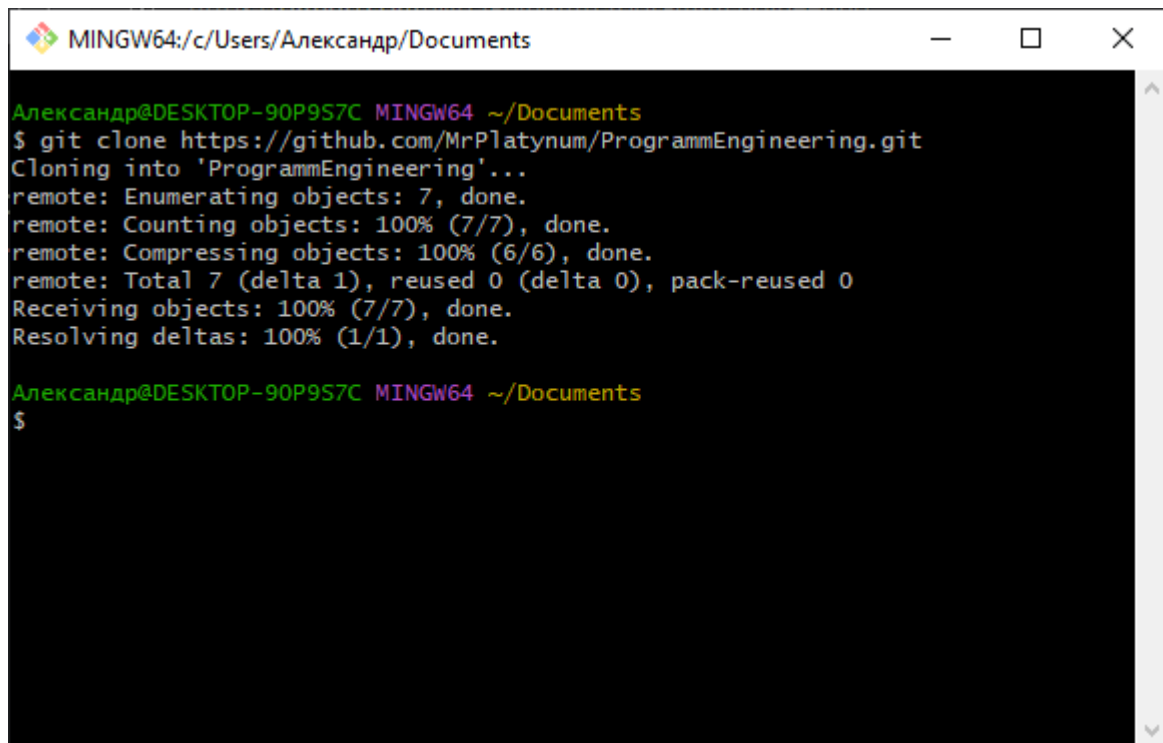


Рисунок 5 – Созданный репозиторий

2. Выполним клонирование созданного репозитория на рабочий компьютер:

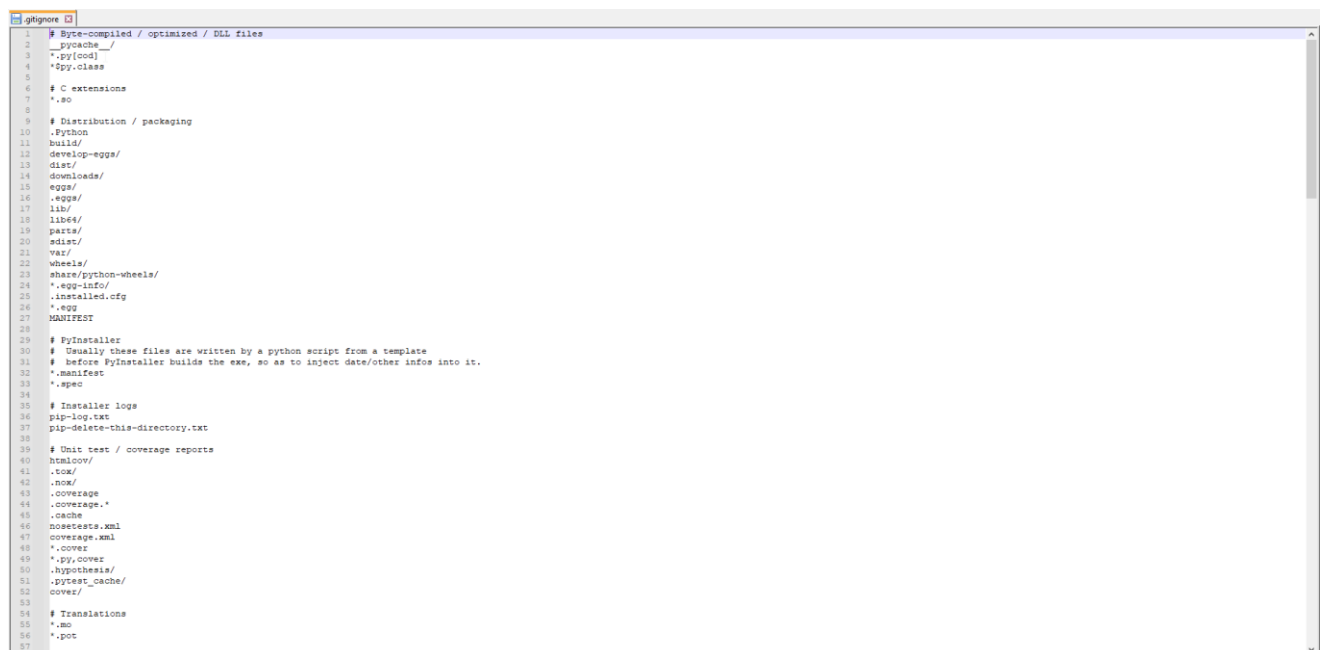


```
MINGW64/c/Users/Александр/Documents
Александр@DESKTOP-90P9S7C MINGW64 ~/Documents
$ git clone https://github.com/MrPlatynum/ProgrammEngineering.git
Cloning into 'ProgrammEngineering'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), done.
Resolving deltas: 100% (1/1), done.

Александр@DESKTOP-90P9S7C MINGW64 ~/Documents
$
```

Рисунок 6 – Клонирование репозитория с помощью команды «git clone»

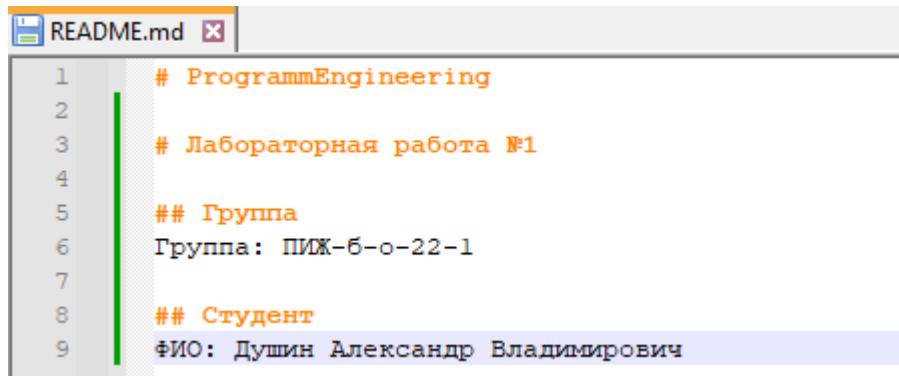
3. Покажем содержимое .gitignore:



```
.gitignore
1 # Byte-compiled / optimized / DLL files
2 __pycache__ /
3 *.py[co]d
4 *.egg-class
5
6 # C extensions
7 *.so
8
9 # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 egg-info/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 *.installed.cfg
26 *.egg
27 MANIFEST
28
29 # PyInstaller
30 # Usually these files are written by a python script from a template
31 # before PyInstaller builds the exe, so as to inject date/other infos into it.
32 *.manifest
33 *.spec
34
35 # Installer logs
36 pip-log.txt
37 pip-delete-this-directory.txt
38
39 # Unit test / coverage reports
40 htmlcov/
41 .tox/
42 .nox/
43 .coverage
44 .coverage.*
45 .cache
46 nosetests.xml
47 coverage.xml
48 *.cover
49 *.py,cover
50 .hypothesis/
51 pytest_cache/
52 cover/
53
54 # Translations
55 *.mo
56 *.pot
57
```

Рисунок 7 – Содержимое файла «.gitignore»

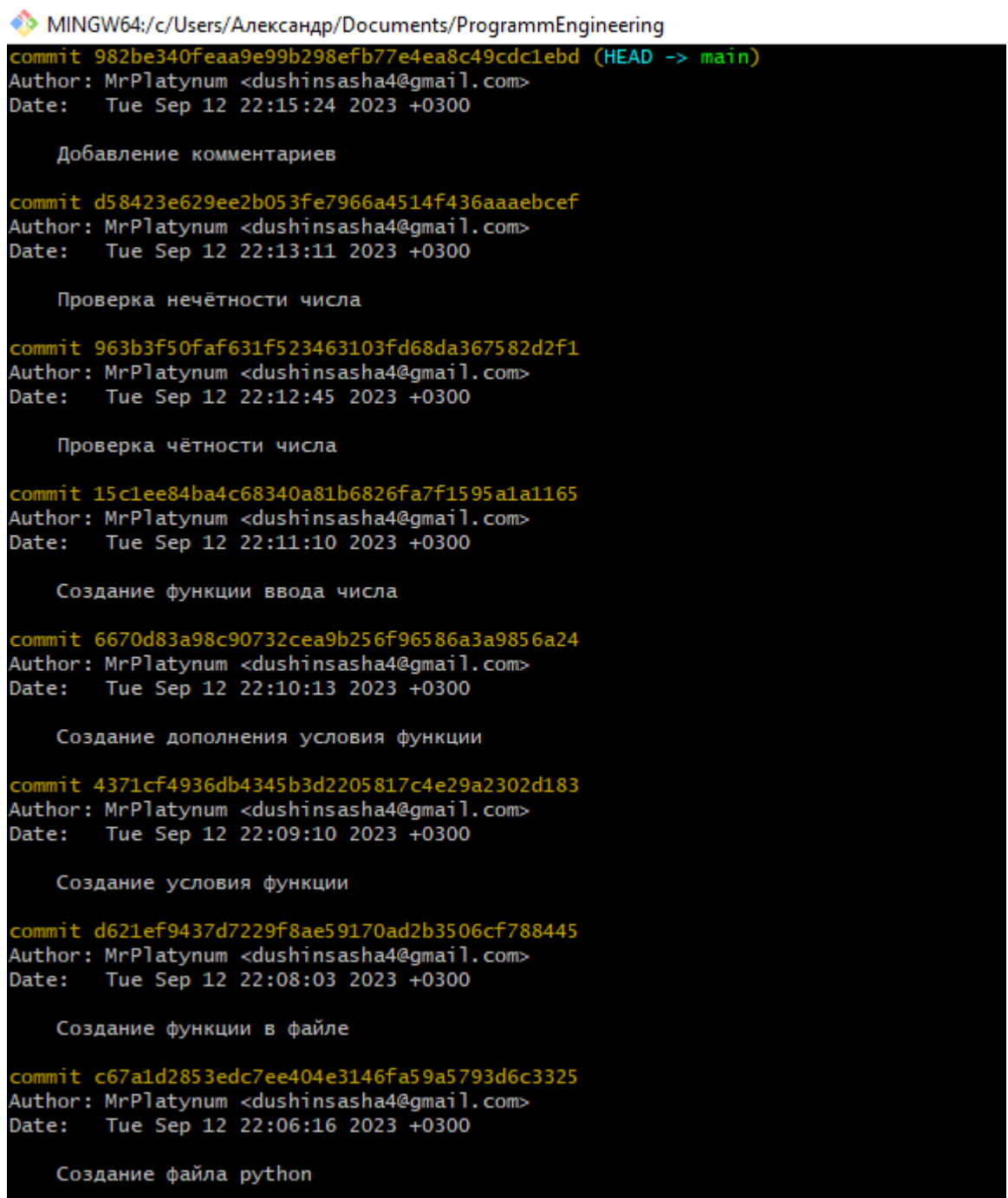
4. Добавим в файл README.md информацию о группе и ФИО студента, выполняющего лабораторную работу:



```
1 # ProgrammEngineering
2
3 # Лабораторная работа №1
4
5 ## Группа
6 Группа: ПИЖ-б-о-22-1
7
8 ## Студент
9 ФИО: Душин Александр Владимирович
```

Рисунок 8 – Добавление информации в файл README.md

5. Напишем небольшую программу на выбранном языке программирования. Зафиксируем изменения в локальном репозитории:



```
MINGW64:/c:/Users/Александр/Documents/ProgrammEngineering
commit 982be340feaa9e99b298efb77e4ea8c49cdc1ebd (HEAD -> main)
Author: MrPlatynum <dushinsasha4@gmail.com>
Date: Tue Sep 12 22:15:24 2023 +0300

    Добавление комментариев

commit d58423e629ee2b053fe7966a4514f436aaebcef
Author: MrPlatynum <dushinsasha4@gmail.com>
Date: Tue Sep 12 22:13:11 2023 +0300

    Проверка нечётности числа

commit 963b3f50faf631f523463103fd68da367582d2f1
Author: MrPlatynum <dushinsasha4@gmail.com>
Date: Tue Sep 12 22:12:45 2023 +0300

    Проверка чётности числа

commit 15c1ee84ba4c68340a81b6826fa7f1595a1a1165
Author: MrPlatynum <dushinsasha4@gmail.com>
Date: Tue Sep 12 22:11:10 2023 +0300

    Создание функции ввода числа

commit 6670d83a98c90732cea9b256f96586a3a9856a24
Author: MrPlatynum <dushinsasha4@gmail.com>
Date: Tue Sep 12 22:10:13 2023 +0300

    Создание дополнения условия функции

commit 4371cf4936db4345b3d2205817c4e29a2302d183
Author: MrPlatynum <dushinsasha4@gmail.com>
Date: Tue Sep 12 22:09:10 2023 +0300

    Создание условия функции

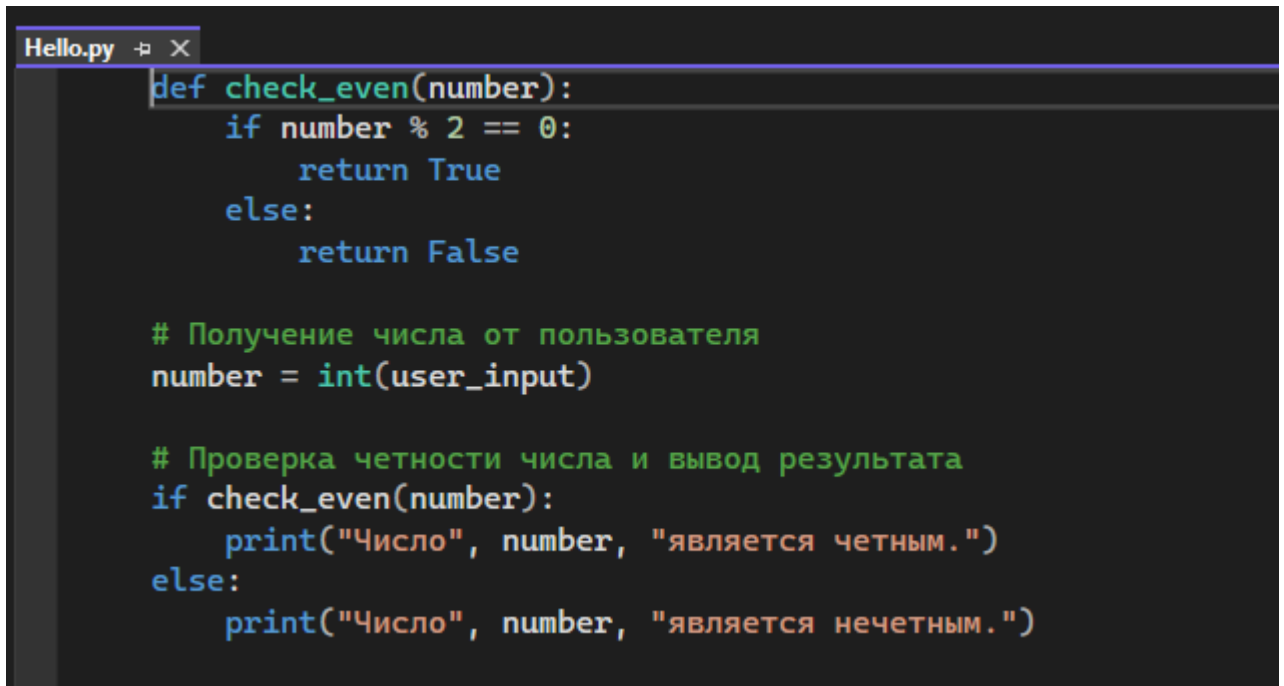
commit d621ef9437d7229f8ae59170ad2b3506cf788445
Author: MrPlatynum <dushinsasha4@gmail.com>
Date: Tue Sep 12 22:08:03 2023 +0300

    Создание функции в файле

commit c67a1d2853edc7ee404e3146fa59a5793d6c3325
Author: MrPlatynum <dushinsasha4@gmail.com>
Date: Tue Sep 12 22:06:16 2023 +0300

    Создание файла python
```

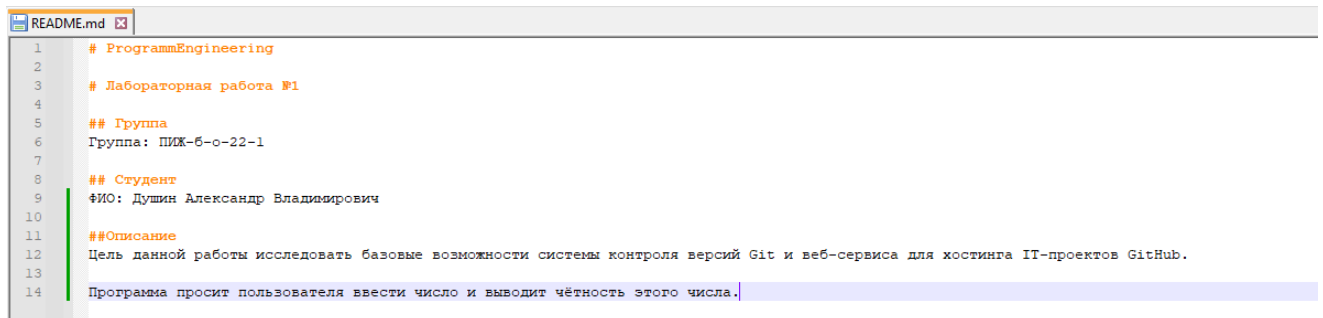
Рисунок 9 – История коммитов



```
def check_even(number):  
    if number % 2 == 0:  
        return True  
    else:  
        return False  
  
# Получение числа от пользователя  
number = int(user_input)  
  
# Проверка четности числа и вывод результата  
if check_even(number):  
    print("Число", number, "является четным.")  
else:  
    print("Число", number, "является нечетным.")
```

Рисунок 10 – Код программы

6. Зафиксируем сделанные изменения в файле README.md:



```
1 # ProgrammEngineering  
2  
3 # Лабораторная работа #1  
4  
5 ## Группа  
6 Группа: ПИЖ-б-о-22-1  
7  
8 ## Студент  
9 ФИО: Душин Александр Владимирович  
10  
11 ## Описание  
12 Цель данной работы исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.  
13  
14 Программа просит пользователя ввести число и выводит чётность этого числа.
```

Рисунок 11 – Итоговый файл README.md

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?

Многие люди в качестве метода контроля версий применяют копирование файлов в отдельную директорию (возможно даже, директорию с отметкой по времени, если они достаточно сообразительны). Данный подход очень распространён из-за его простоты, однако он невероятно сильно подвержен появлению ошибок. Можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели.

Самый очевидный минус — это единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками. Если жёсткий диск, на котором хранится центральная БД, повреждён, а своевременные бэкапы отсутствуют, вы потеряете всё — всю историю проекта, не считая единичных снимков репозитория, которые сохранились на локальных машинах разработчиков. Локальные СКВ страдают от той же самой проблемы: когда вся история проекта хранится в одном месте, вы рискуете потерять всё.

3. К какой СКВ относится Git?

Git является распределенной СКВ.

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 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. Вместо этого, подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.

5. Как обеспечивается целостность хранимых данных в Git?

Git обеспечивает целостность хранимых данных с помощью хеш-контрольных сумм. Каждый объект в Git, будь то файл, коммит или ветка, имеет уникальный хеш. При сохранении объекта в репозитории, Git вычисляет его хеш и сохраняет объект под этим хешем. При последующей загрузке объекта, Git снова вычисляет хеш и сравнивает его с сохраненным. Если хеши совпадают, значит данные не были изменены и сохранены верно. Если хеши не совпадают, это указывает на изменения данных или на ошибку в системе хранения. Кроме того, Git использует дерево объектов для обеспечения целостности данных. Дерево объектов представляет собой графическую структуру, где каждый коммит указывает на предшествующие коммиты, файлы и другие объекты. Все коммиты и объекты в дереве хранятся с использованием их хешей, что обеспечивает целостность данных.

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

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

Если определённая версия файла есть в Git-директории, эта версия считается зафиксированной.

Если версия файла изменена и добавлена в индекс, значит, она подготовлена. И если файл был изменён с момента последнего

распаковывания из репозитория, но не был добавлен в индекс, он считается изменённым.

7. Что такое профиль пользователя в GitHub?

Профиль - это ваша публичная страница на GitHub, как и в социальных сетях. Когда вы ищете работу в качестве программиста, работодатели могут посмотреть ваш профиль GitHub и принять его во внимание, когда будут решать, брать вас на работу или нет.

8. Какие бывают репозитории в GitHub?

Локальный — расположен на одном компьютере, и работать с ним может только один человек.

Централизованный — расположен на сервере, куда имеют доступ сразу несколько программистов.

Распределенный — самый удобный вариант с облачным

9. Укажите основные этапы модели работы с GitHub.

Сначала рассмотрим область GitHub. В нем есть два хранилища: upstream - это оригинальный репозиторий проекта, который вы скопировали, origin - ваш fork (копия) на GitHub, к которому у вас есть полный доступ. Чтобы перенести изменения с вашей копии в исходному репозиторий проекта, вам нужно сделать запрос на извлечение. Если вы хотите внести небольшие изменения в свою копию (fork), вы можете использовать вебинтерфейс GitHub. Однако такой подход не удобен при разработке программ, поскольку вам часто приходится запускать и отлаживать их локально. Стандартный способ - создать локальный клон удаленного репозитория и работать с ним локально, периодически внося изменения в удаленный репозиторий.

10. Как осуществляется первоначальная настройка Git после установки?

Чтобы убедиться, что Git был успешно установлен, введите команду ниже в терминале, чтобы отобразить текущую версию вашего Git:

```
git version
```

Если она сработала, давайте добавим в настройки Git ваше имя, фамилию и адрес электронной почты, связанный с вашей учетной записью

GitHub:

```
git config --global user.namegit config --global user.email
```

11. Опишите этапы создания репозитория в GitHub.

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?

GitHub поддерживает разнообразные типы лицензий, включая MIT License, GNU GPL, Apache License, Creative Commons Licenses, Unlicense и другие.

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

После создания репозитория его необходимо клонировать на ваш компьютер. Для этого на странице репозитория необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования.

Откройте командную строку или терминал и перейдите в каталог, куда вы хотите скопировать хранилище. Затем напишите git clone и введите адрес:

```
git clone https://github.com/kushedow/flask-html
```

14. Как проверить состояние локального репозитория Git?

Локальный репозиторий включает в себя все те же файлы, ветки и историю коммитов, как и удаленный репозиторий. Введите эту команду, чтобы проверить состояние вашего репозитория:

```
git status
```

15. Как изменяется состояние локального репозитория Git после выполнения следующих операций:

Команда `git add` добавляет изменение из рабочего каталога в раздел проиндексированных файлов. Она сообщает Git, что вы хотите включить изменения в конкретном файле в следующий коммит.

Команда `git commit` сохраняет снимок состояния из раздела проиндексированных файлов в истории коммитов репозитория.

Важно, что коммит добавляет изменения только в ваш локальный репозиторий. Если вы хотите распространить их в исходный репозиторий на GitHub, вам нужно использовать `push`. В первый раз вам также необходимо отправить свою локальную ветку, потому что она не существует в удаленном репозитории.

```
git push --set-upstream origin edit-readme
```

В следующий раз сделать это будет уже проще:

```
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`.

Откройте терминал или командную строку на каждом из рабочих компьютеров.

На каждом компьютере перейдите в папку, где хотите сохранить локальную копию репозитория.

Используйте команду `git clone <url репозитория>` для клонирования репозитория с GitHub на оба компьютера. Например: `git clone https://github.com/username/repo.git`.

Перейдите в папку репозитория на одном из компьютеров с помощью команды `cd repo`.

Добавьте ссылку на второй компьютер в качестве удаленного репозитория на первом компьютере с помощью команды `git remote add <имя_удаленного_репо> <url_второго_компьютера>`. Например: `git remote add second_computer <url_второго_компьютера>`.

Синхронизируйте локальный репозиторий с удаленным репозиторием на GitHub на обоих компьютерах, используя команду `git pull <имя_удаленного_репо> <имя_ветки>`. Например: `git pull origin master`. При необходимости выполните эту команду на обоих компьютерах.

Теперь оба локальных репозитория, связанные с репозиторием на GitHub, будут находиться в синхронизированном состоянии. Вы можете выполнять работу над проектом на любом из компьютеров и загружать изменения на GitHub с помощью команды `git push <имя_удаленного_репо> <имя_ветки>`. Например: `git push origin master`. И другой компьютер может получить эти изменения, выполнив команду `git pull <имя_удаленного_репо> <имя_ветки>`. Например: `git pull origin master`.

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

GitHub и GitLab - популярные сервисы для работы с Git. GitHub широко используется с большим сообществом, интеграцией инструментов и ограниченными бесплатными возможностями для частных репозиториев. GitLab предоставляет мощные инструменты CI/CD, самостоятельное развертывание и более функциональные бесплатные версии для частных репозиториев. Выбор зависит от потребностей и предпочтений.

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.

Некоторые GUI-средства для Git включают GitHub Desktop, GitKraken, SourceTree и TortoiseGit. С их помощью можно удобно выполнить операции Git, такие как клонирование репозитория, создание коммитов и отправка изменений на удаленный репозиторий.