Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
дисциплины «Основы программной инженерии»

Выполнила:
Рядская Мария Александровна
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р. А., доцент кафедры
инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Ход работы

1. Я изучила теоретический материал работы

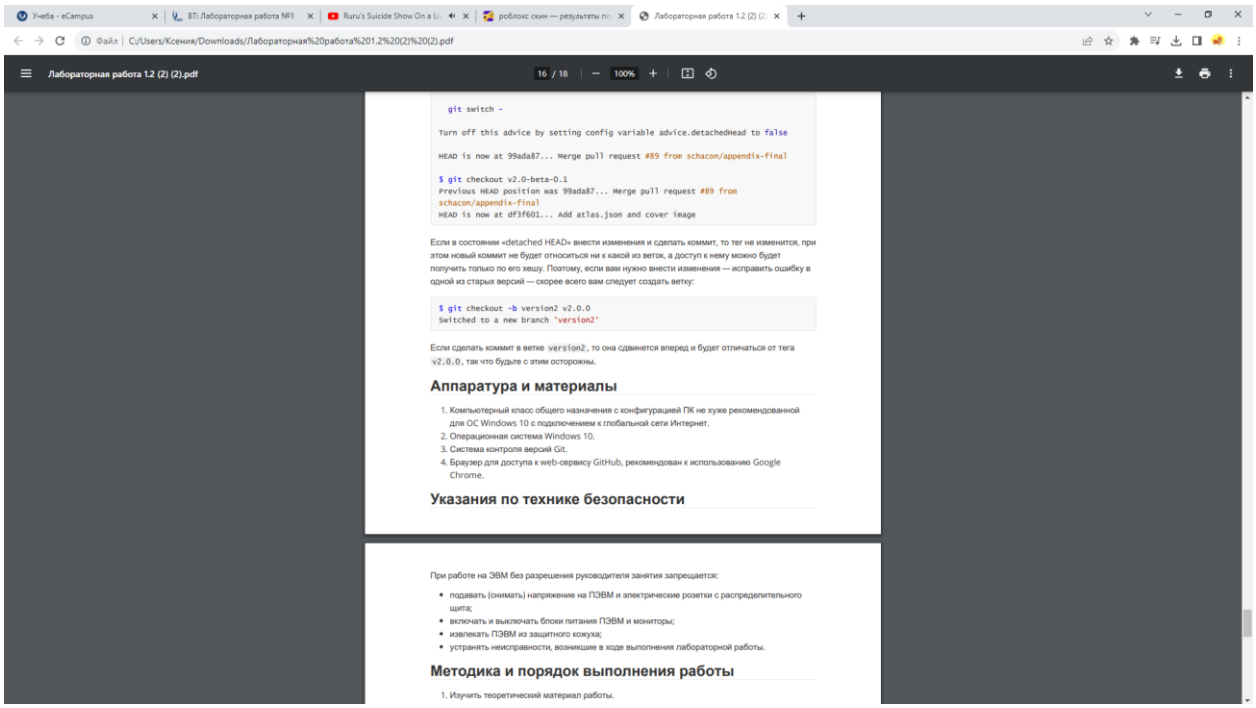


Рисунок 1.1 – Изучение материала для лабораторной работы

2. Создала общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный мною язык программирования

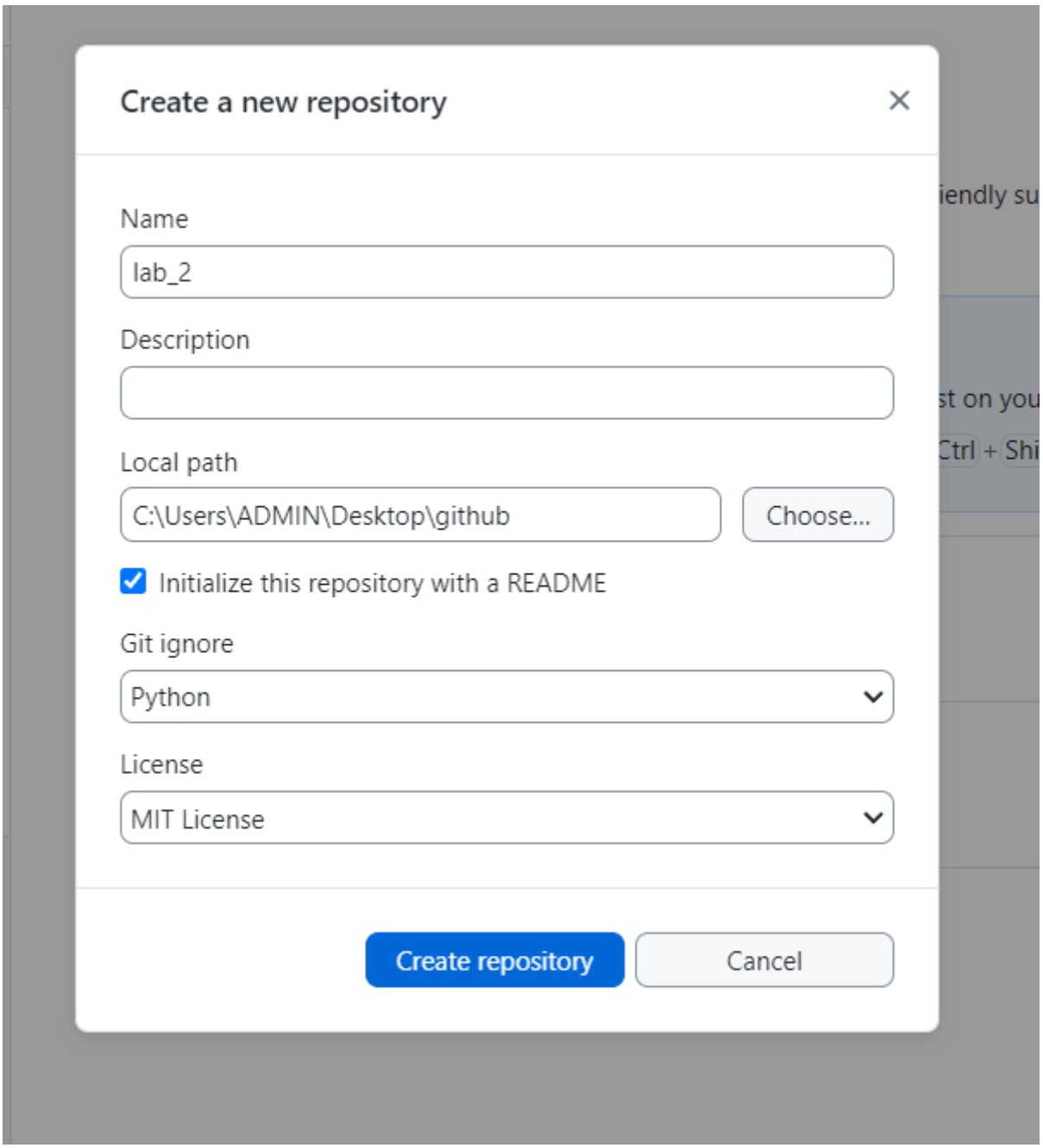


Рисунок 2.1 – Настройка репозитория

3. Проработала примеры лабораторной работы

```
C:\git1\PJ>git commit -m"programm"
[main f2874be] programm
 2 files changed, 9 insertions(+), 2 deletions(-)
 delete mode 100644 1
 create mode 100644 programm.txt

C:\git1\PJ>git push
fatal: User canceled device code authentication
Username for 'https://github.com': git push
Password for 'https://git push@github.com':
remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/mryadskaya/PJ.git/'

C:\git1\PJ>\git push
"\git" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.

C:\git1\PJ>git push
info: please complete authentication in your browser...
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 421 bytes | 105.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/mryadskaya/PJ.git
   9e2a3cd..f2874be  main -> main

C:\git1\PJ>
```

1. Выполнила клонирование созданного репозитория на рабочий компьютер

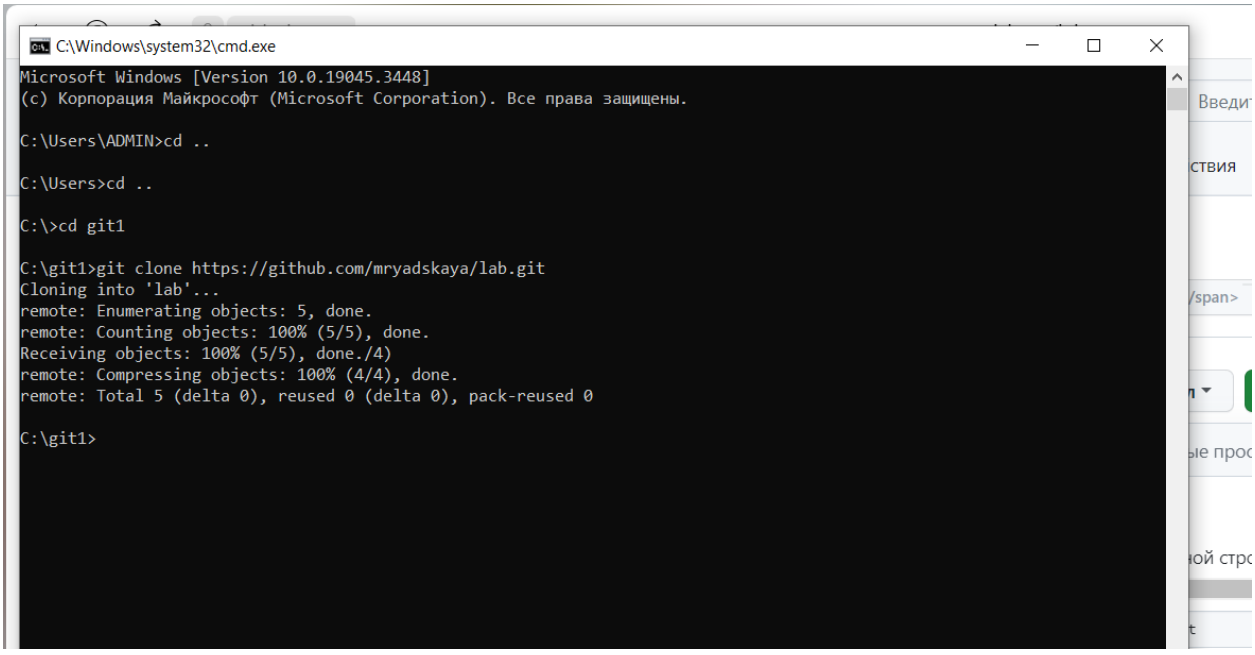


Рисунок 3.1 – Копирование ссылки репозитория

4. Дополнила файл .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки

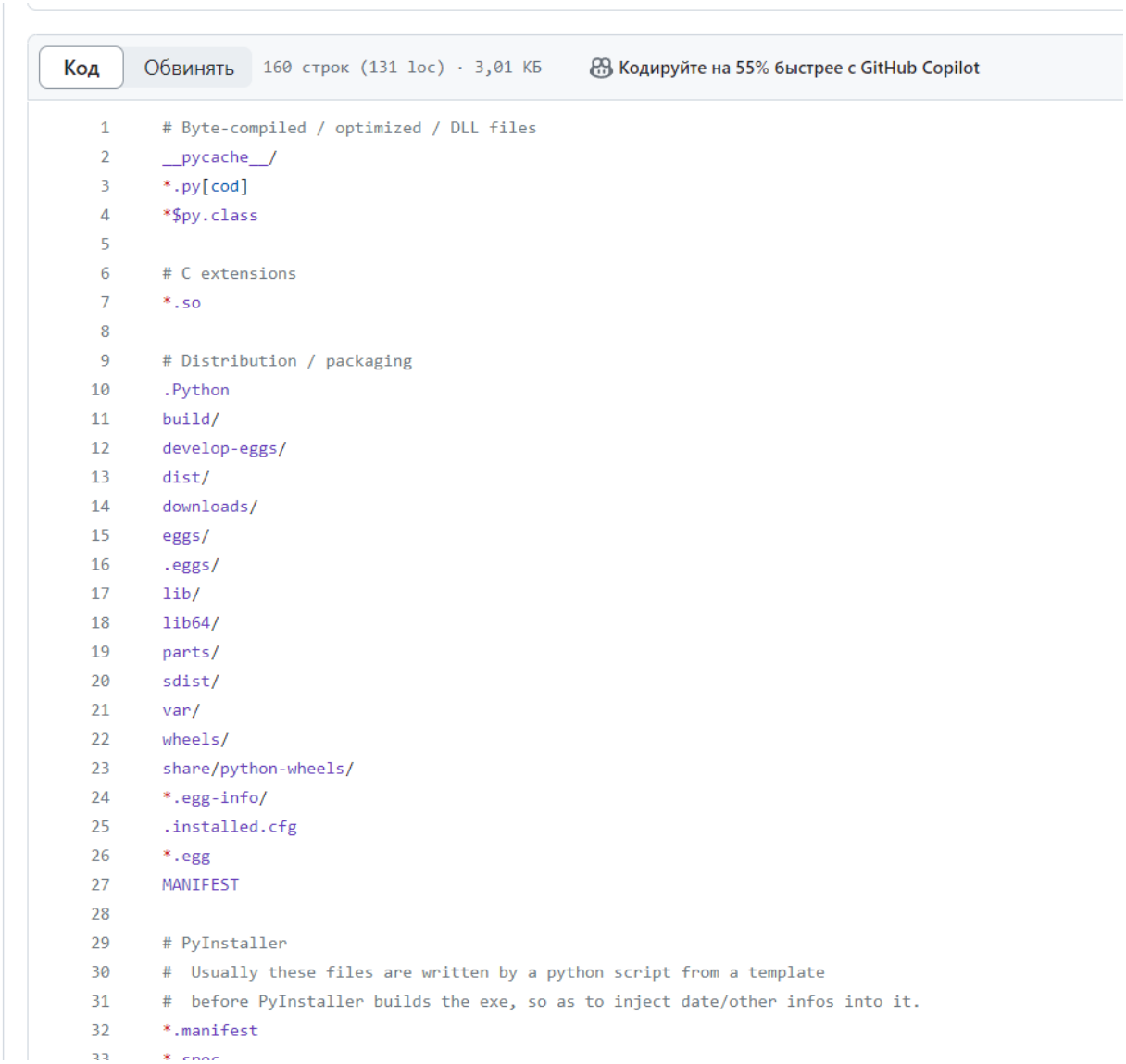


Рисунок 41 – Файл с необходимыми правилами для языка Python

5. Добавила в файл README.md информацию о своей группе и моём ФИО

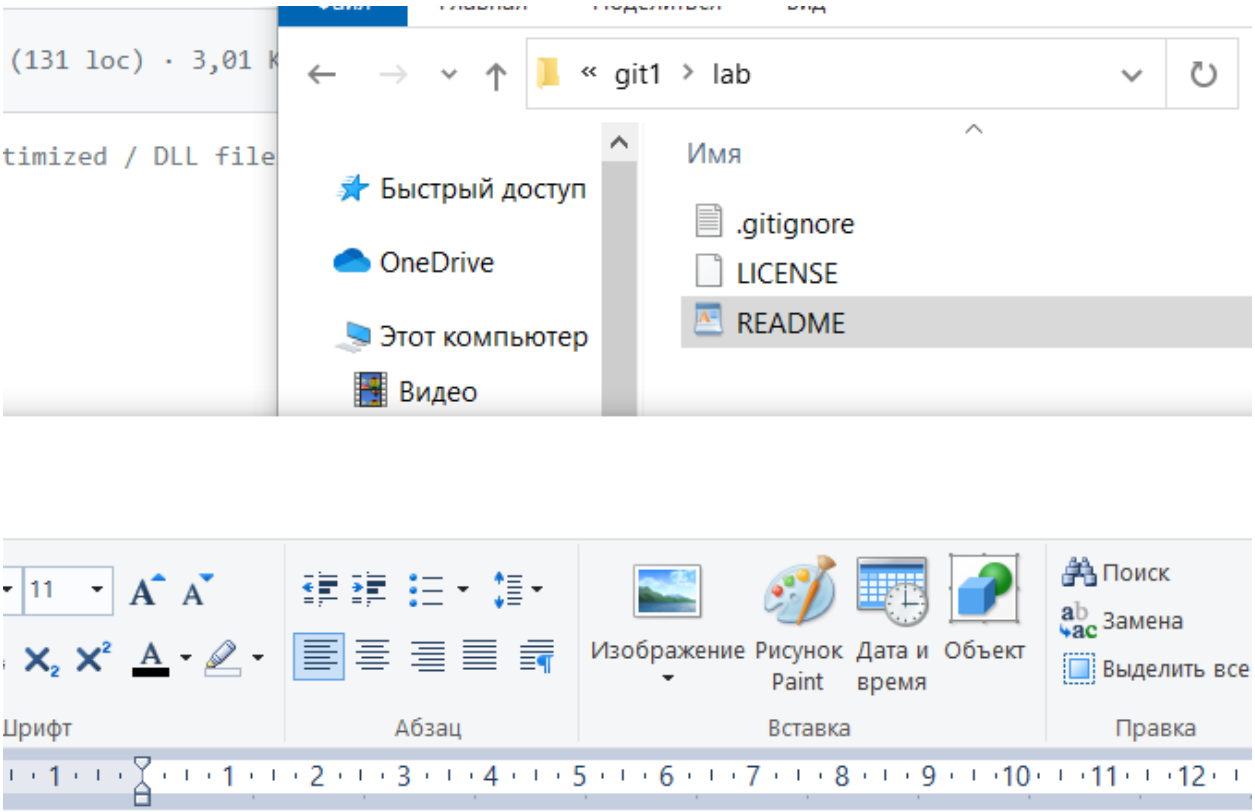


Рисунок 5.1 – Локально создаю файл README.md

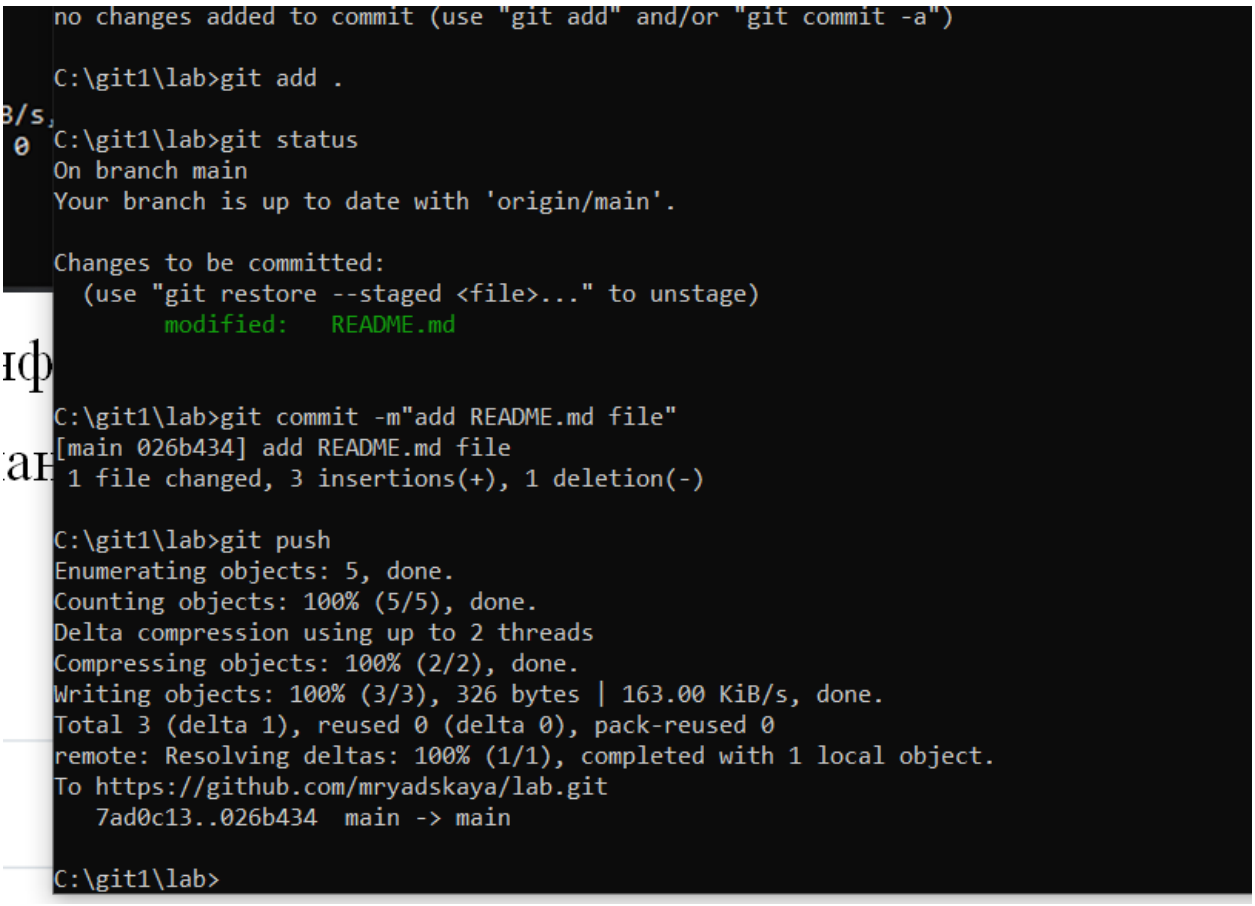


Рисунок 5.2 – С помощью командной строки пересылаю обновленный репозиторий на GitHub

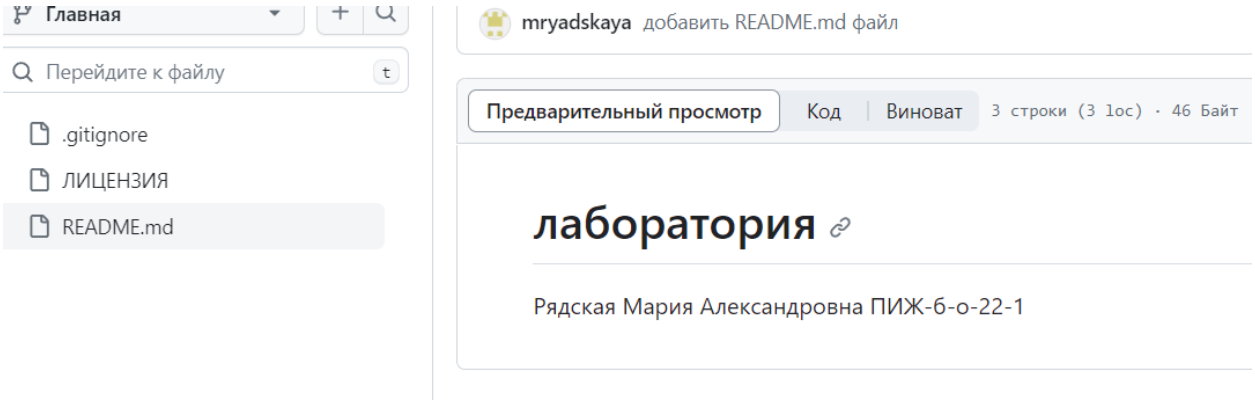


Рисунок 5.3 – Мой файл README.md на GitHub

6. Написала небольшую программу на выбранном мною языке программирования. Зафиксировала изменения при написании программы в локальном репозитории. Сделала не менее 7 коммитов, отмеченных не менее 3 тэгами

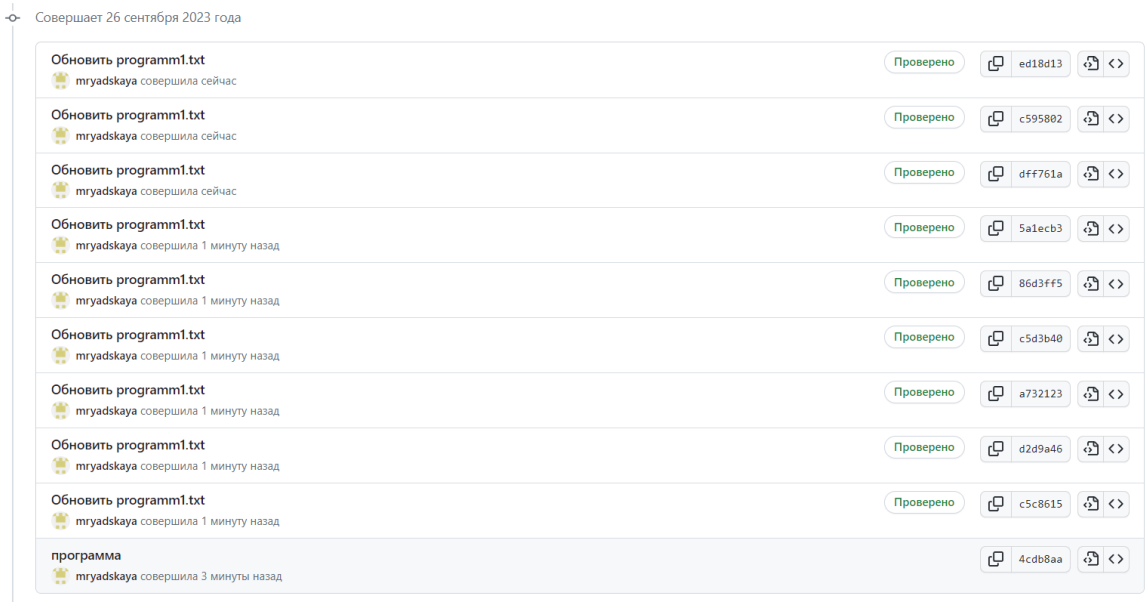


Рисунок 6 – Изменения, отображённые на GitHub

```
C:\git1\lab>git tag -a v1.3 -m "my version 1.3"

C:\git1\lab>git tag
v
v1.0
v1.3
v1.4
v2.0
v3.0

C:\git1\lab>git push origin --tags
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 2 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (13/13), 3.46 KiB | 168.00 KiB/s, done.
Total 13 (delta 3), reused 4 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/mryadskaya/lab.git
 * [new tag]         v -> v
 * [new tag]         v1.0 -> v1.0
 * [new tag]         v1.3 -> v1.3
 * [new tag]         v1.4 -> v1.4
 * [new tag]         v2.0 -> v2.0
 * [new tag]         v3.0 -> v3.0
```

Рисунок 7 – Отмечаю не менее 3 тэгов

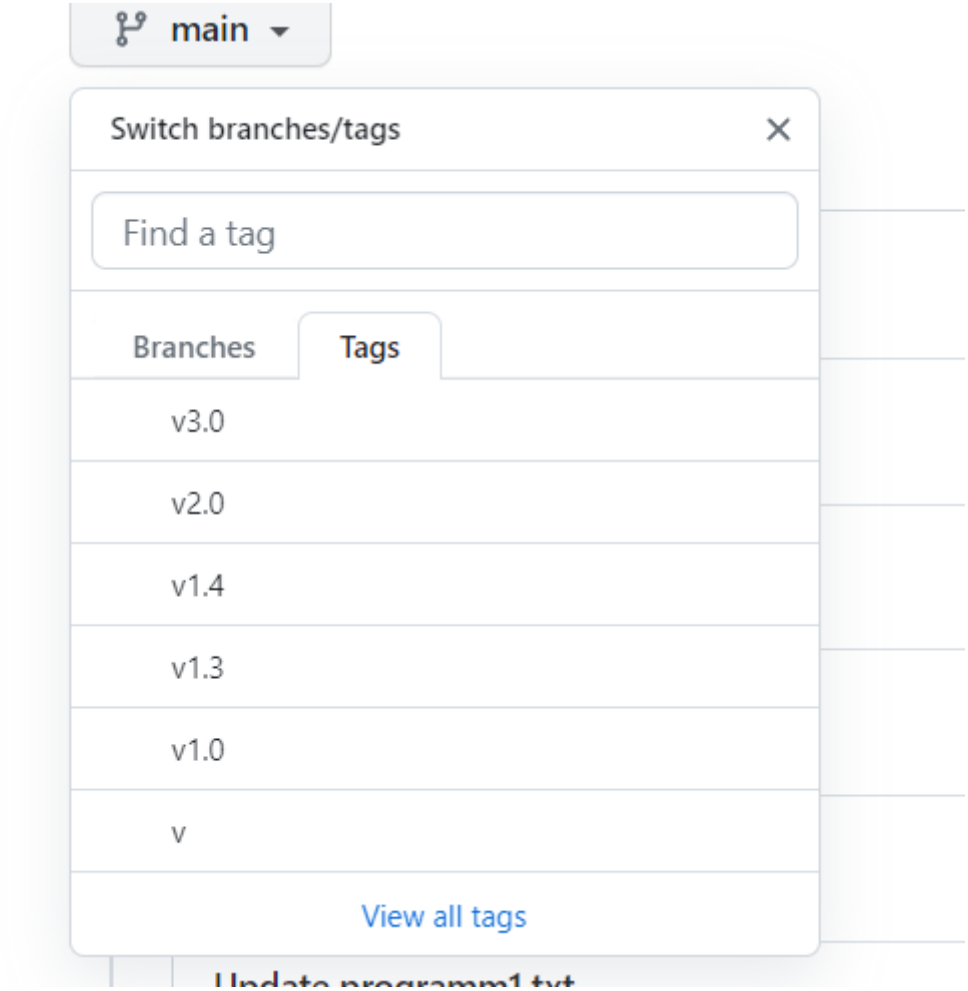


Рисунок 7.1 – Отмеченные тэги, отображённые на GitHub

7. Просмотрела историю (журнал) хранилища командой gitlog с помощью команды gitlog --graph --pretty=oneline --abbrev-commit

```
fatal: unrecognized argument: --abbrev-commit

C:\git1\lab>git log --graph --pretty=oneline --abbrev-commit
* 4cdb8aa (HEAD -> main, tag: v3.0, tag: v2.0, tag: v1.4, tag: v1.3, tag: v1.0, tag: v, origin/main, origin/HEAD) prigra
mm
* 026b434 add README.md file
* 7ad0c13 Initial commit

C:\git1\lab>
```

Рисунок 8 – История хранилища

1. Просмотреть содержимое коммитов командой gitshowHEAD, gitshowHEAD~1, gitshowHEAD

Рисунок 9 – Вывод последнего коммита с помощью команды gitshowHEAD

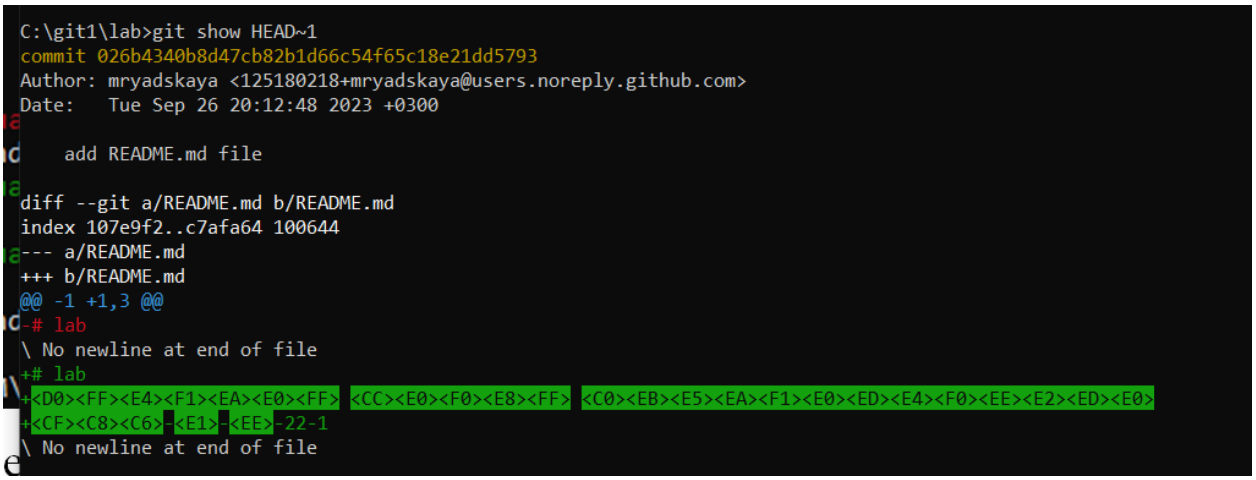
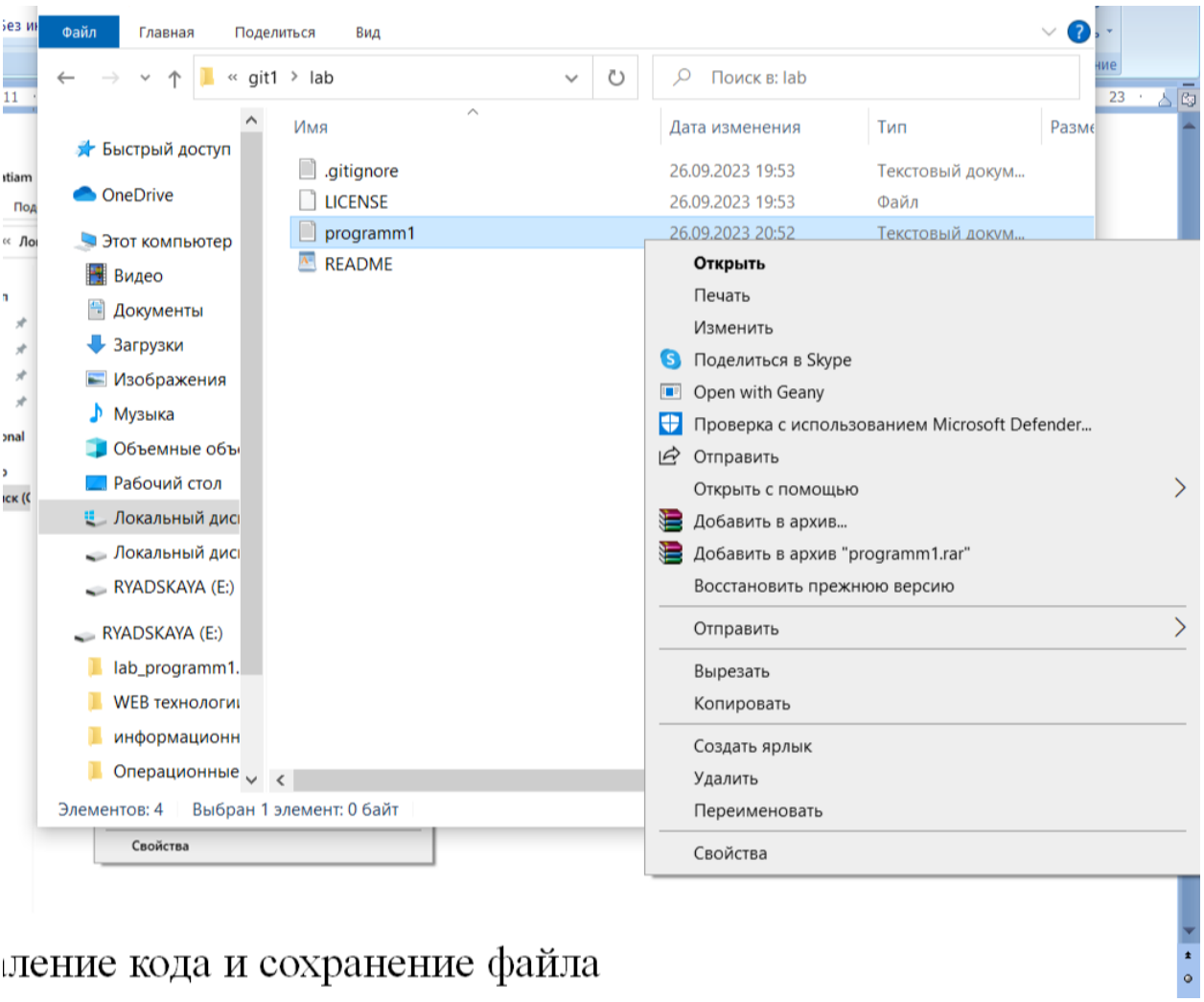


Рисунок 9.2 – Вывод предпоследнего коммита с помощью команды gitshowHEAD~1

1. Я освоила возможность отката к заданной версии

   1.1. Удалила весь код из одного из файлов программы репозитория, в моём случаеIamyour.py, и сохранила этот файл.



ление кода и сохранение файла

Рисунок 10.1– Удаление кода и сохранение файла

7.1. Удалила все несохраненные изменения в файле командой:gitcheckout–Iamyour.py

```
C:\git1\lab>git checkout -- programm.txt
error: pathspec 'programm.txt' did not match any file(s) known to git

C:\git1\lab>git checkout -- programm1.txt

C:\git1\lab>
```

Рисунок 10.2 – Восстановление последнего коммита с помощью команды gitcheckout–Iamyour.py

1.1. Повторяю пункт 10.1 и делаю коммит

```
C:\git1\lab>git push
Everything up-to-date

C:\git1\lab>git checkout -- programm1.txt

C:\git1\lab>git add .

C:\git1\lab>git commit -m"accidentally deleted my program"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\git1\lab>git commit -m"programm1"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\git1\lab>git push
Everything up-to-date

C:\git1\lab>
```

Рисунок 10.3 – Отправляю удалённый файл на GitHub

1.1. Откатываю состояние хранилища к предыдущей версии командой: gitreset --hard HEAD~1

```
C:\git1\lab>git reset --hard HEAD~1
HEAD is now at 4576410 program1

C:\git1\lab>
```

Рисунок 10.4 – Возвращаем предпоследний коммит

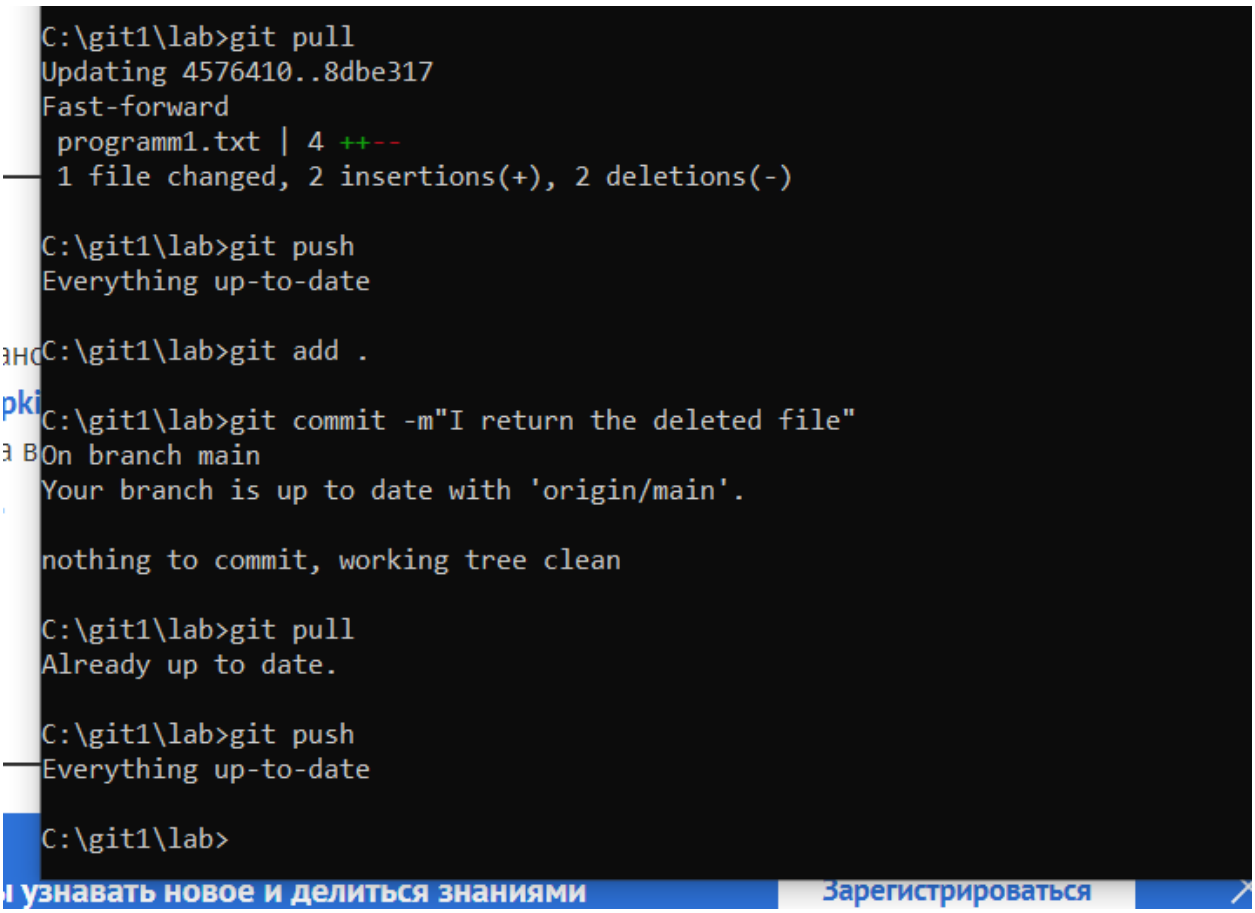
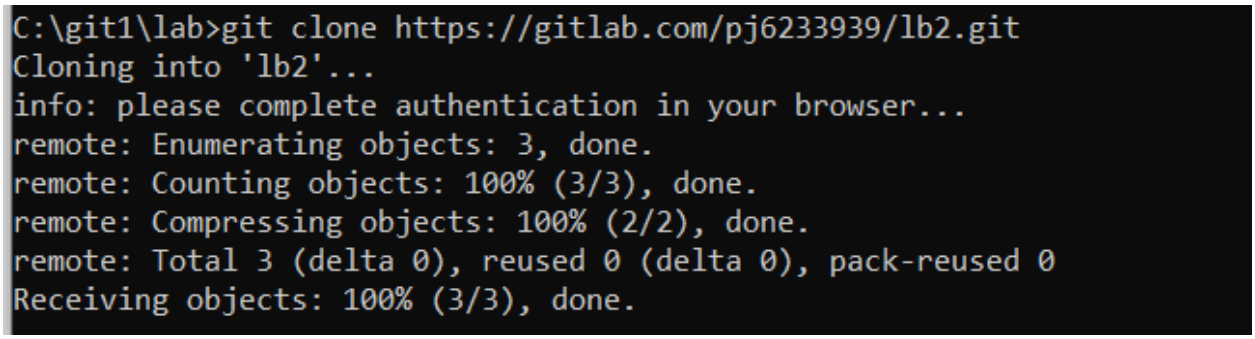


Рисунок 10.4.1 – Возвращаем предпоследний коммит

Вывод:Gitпредоставляет возможность отката к предыдущей версии. Это позволяет исправить ошибки или проблемы, возникшие после обновления кода, а также сохранять историю изменений, что облегчает восстановление предыдущих версий проекта при необходимости. С помощью команды gitcheckout -- <имя файла>можно вернуть файл к последнему фиксированному состоянию, а с помощью командыgitreset --hard HEAD~1 откатить состояние хранилища к предыдущей версии. То есть даже после выполнения коммита можно вернуть предыдущую версию программы.



```
C:\git1\lab>git add .
warning: adding embedded git repository: lb2
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> lb2
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached lb2
hint:
hint: See "git help submodule" for more information.

C:\git1\lab>git commit -m"pr2"
[main c4f3b01] pr2
 1 file changed, 1 insertion(+)
 create mode 160000 lb2

C:\git1\lab>git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 266 bytes | 66.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/mryadskaya/lab.git
   8dbe317..c4f3b01  main -> main
```

```
C:\git1\lab>git fetch --prune

C:\git1\lab>git push --prune git@gitlab.com:pj6233939/lb2.git
The authenticity of host 'gitlab.com (172.65.251.78)' can't be established.
ED25519 key fingerprint is SHA256:eUXGGm1YGsMAS7vkcx6JOJdOGHPem5gQp4taiCfCLB8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

```
C:\git1\lab>git fetch --prune
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 1001.82 KiB | 487.00 KiB/s, done.
From https://github.com/mryadskaya/lab
   c4f3b01..5a4dbf9  main       -> origin/main

C:\git1\lab>git push git@gitlab.com:pj6233939/2.git +refs/remotes/origin/*:refs/heads/* +refs/tags/*:refs/tags/*
The authenticity of host 'gitlab.com (172.65.251.78)' can't be established.
ED25519 key fingerprint is SHA256:eUXGGm1YGsMAS7vkcx6JOJdOGHPem5gQp4taiCfCLB8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'gitlab.com' (ED25519) to the list of known hosts.
Connection closed by 172.65.251.78 port 22
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

```
C:\git1>cd lb

C:\git1\lb>git clonehttps://github.com/mryadskaya/lab.git
git: 'clonehttps://github.com/mryadskaya/lab.git' is not a git command. See 'git --help'.

C:\git1\lb>git clone https://github.com/mryadskaya/lab.git
Cloning into 'lab'...
remote: Enumerating objects: 53, done.
remote: Counting objects: 100% (53/53), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 53 (delta 28), reused 20 (delta 8), pack-reused 0Receiving objects:  75% (40/53), 972.00 KiB | 131.00 KiB/R

Resolving deltas: 100% (28/28), done.

C:\git1\lb>git fetch --prune
fatal: not a git repository (or any of the parent directories): .git

C:\git1\lb>cd lab

C:\git1\lb\lab>git fetch --prune

C:\git1\lb\lab>git push --prune git@bitbucket.org:pj-lab-ryadskaya/lab2.git +refs/remotes/origin/*:refs/heads/* +refs/tags/*:refs/tags/*
git@bitbucket.org: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

Контрольные вопросы

1. Как выполнить историю коммитов в Git? Какие существуют дополнительные опции для просмотра истории коммитов?

Одним из основных и наиболее мощных инструментов для просмотра истории коммитов является команда gitlog. Команда gitlog имеет очень большое количество опций для поиска коммитов по разным критериям. Одним из самых полезных аргументов является -p или --patch, который показывает разницу, внесенную в каждый коммит. Если вы хотите увидеть сокращенную статистику для каждого коммита, вы можете использовать опцию –stat. Наиболее интересной опцией является format, которая позволяет указать формат для вывода информации

2. Как ограничить вывод при просмотре истории коммитов?

В дополнение к опциям форматирования вывода команда gitlog принимает несколько опций для ограничения вывода — опций, с помощью которых можно увидеть определенное подмножество коммитов. Вы можете использовать –<n> , где n — это любое натуральное число и представляет собой n последних коммитов. Опции для ограничения вывода по времени, такие как --since и --until, являются очень удобными.Опция --author дает возможность фильтровать по автору коммита, а опция --grep искать по ключевым словам в сообщении коммита.

3. Как внести изменения в уже сделанный коммит?

Если вы хотите переделать коммит — внесите необходимые изменения, добавьте их в индекс и сделайте коммит ещё раз, указав параметр --amend

4. Как отменить индексацию файла в Git?

Используйте gitreset HEAD <имя_файла>… для отмены индексации файла

5. Как отменить изменения в файле?

С помощью команды gitcheckout -- <имя_файла>. Она выполнит откат изменений

6. Что такое удаленный репозиторийGit?

Удалённые репозитории представляют собой версии вашего проекта, сохранённые в интернете или ещё где-то в сети

7. Как выполнить просмотр удаленных репозиториев данного локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду gitremote. Она выведет названия доступных удалённых репозиториев.

8. Как добавить удаленный репозиторий для данного локального репозитория?

Для того, чтобы добавить удалённый репозиторий и присвоить ему имя (shortname), просто выполните команду gitremoteadd<shortname><url>

9. Как выполнить отправку/получение изменений с удаленного репозитория?

Отправка изменений с удалённого репозитория осуществляется с помощью команды gitpush, а получение изменений с удаленного репозитория – с помощью команды gitfetchили gitpull

10. Как выполнить просмотр удаленного репозитория?

Если хотите получить побольше информации об одном из удалённых репозиториев, вы можете использовать команду gitremoteshow<remote>

11. Каково назначение тэгов Git?

Эта функциональность используется для отметки моментов выпуска версий

12. Как осуществляется работа с тэгами Git?

Git использует два основных типа тегов: легковесные и аннотированные. Легковесный тег — это что-то очень похожее на ветку, которая не изменяется — просто указатель на определённый коммит. А вот аннотированные теги хранятся в базе данных Git как полноценные объекты. Создание аннотированного тега в Git выполняется легко. Самый простой способ — это указать - a при выполнении команды tag. Опция -m задаёт сообщение, которое будет храниться вместе с тегом. По умолчанию, команда gitpush не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на удалённый сервер. Процесс аналогичен отправке веток — достаточно выполнить команду gitpushorigin<tagname>. Если у вас много тегов, и вам хотелось бы отправить все за один раз, то можно использовать опцию --tags для команды gitpush. Для удаления тега в локальном репозитории достаточно выполнить команду gittag -d.

13. Самостоятельно изучите назначение флага --prune в командах gitfetch и gitpush. Каково назначение этого флага?

Команда --pruneвводится для очистки любых устаревших или «зависших» ссылок в локальном репозитории. Сюда входят объекты, которые больше недоступны из какой-либо ветви или тега, а также ветви

удалённого отслеживания, которые больше не существуют на удалённом компьютере.