

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: <u>Воронкин Р. А., доцент кафедры инфокоммуникаций</u> _____ (подпись)

Отчет защищен с оценкой _____	Дата защиты _____
-------------------------------	-------------------

Ставрополь, 2023 г.

Ход работы

- Я изучила теоретический материал работы

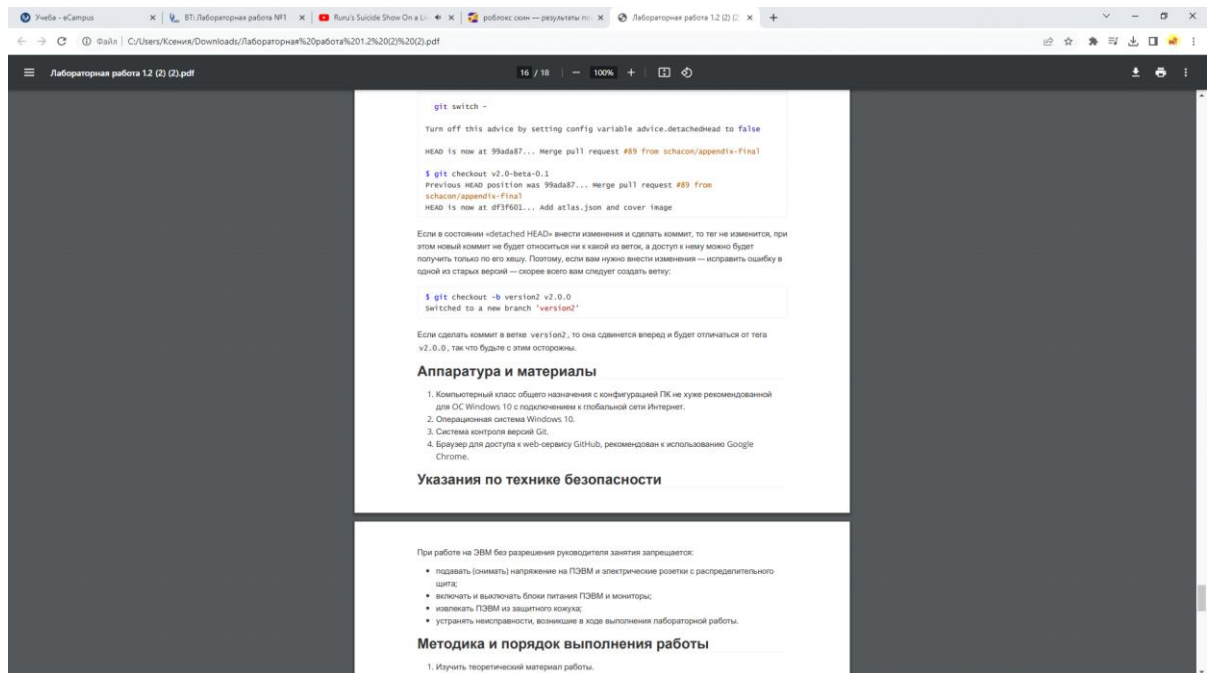


Рисунок 1.1 – Изучение материала для лабораторной работы

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

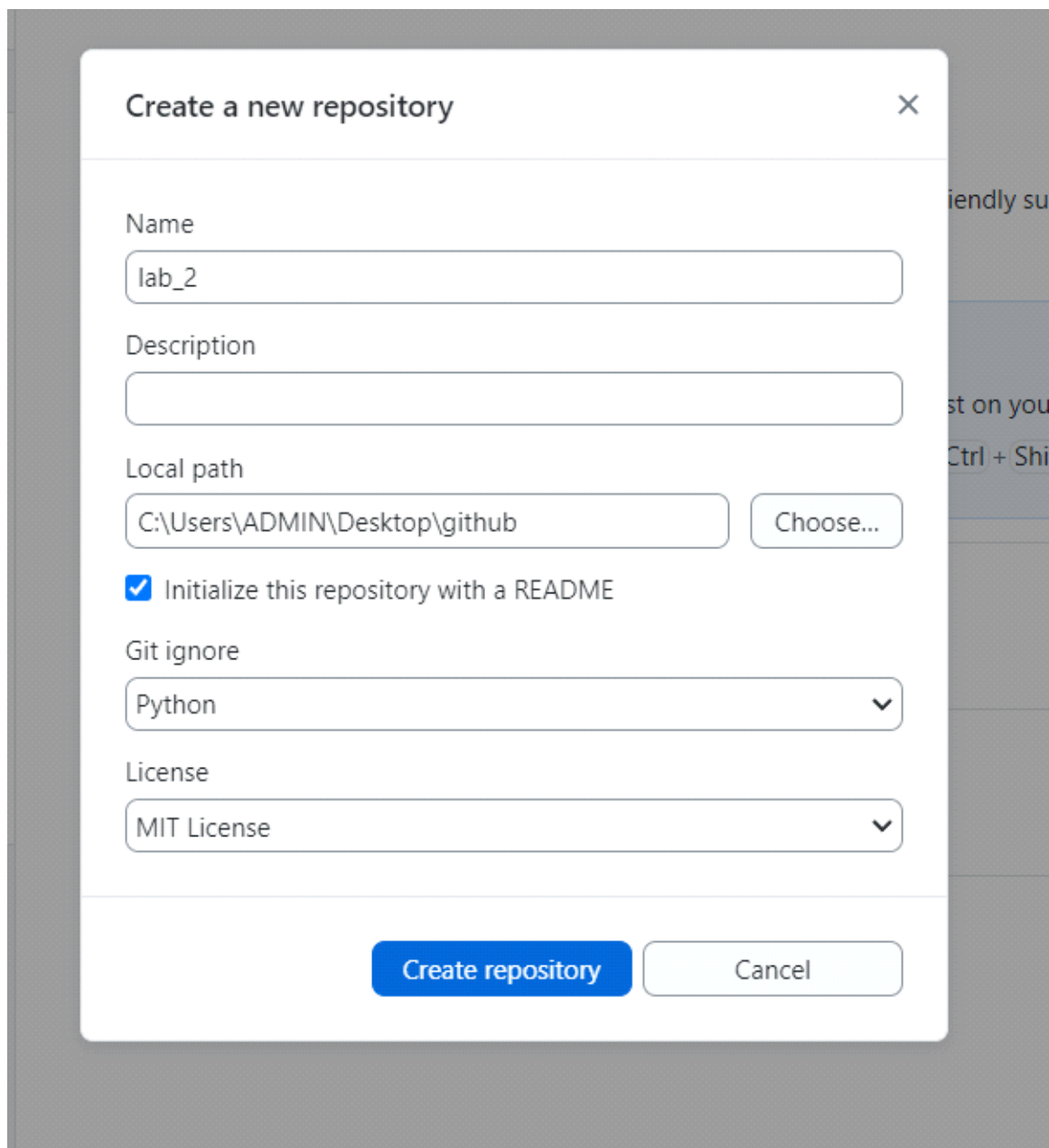


Рисунок 2.1 – Настройка репозитория

- 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>

```

- Выполнила клонирование созданного репозитория на рабочий компьютер

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3448]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\ADMIN>cd ..
C:\Users>cd ..
C:\>cd git1

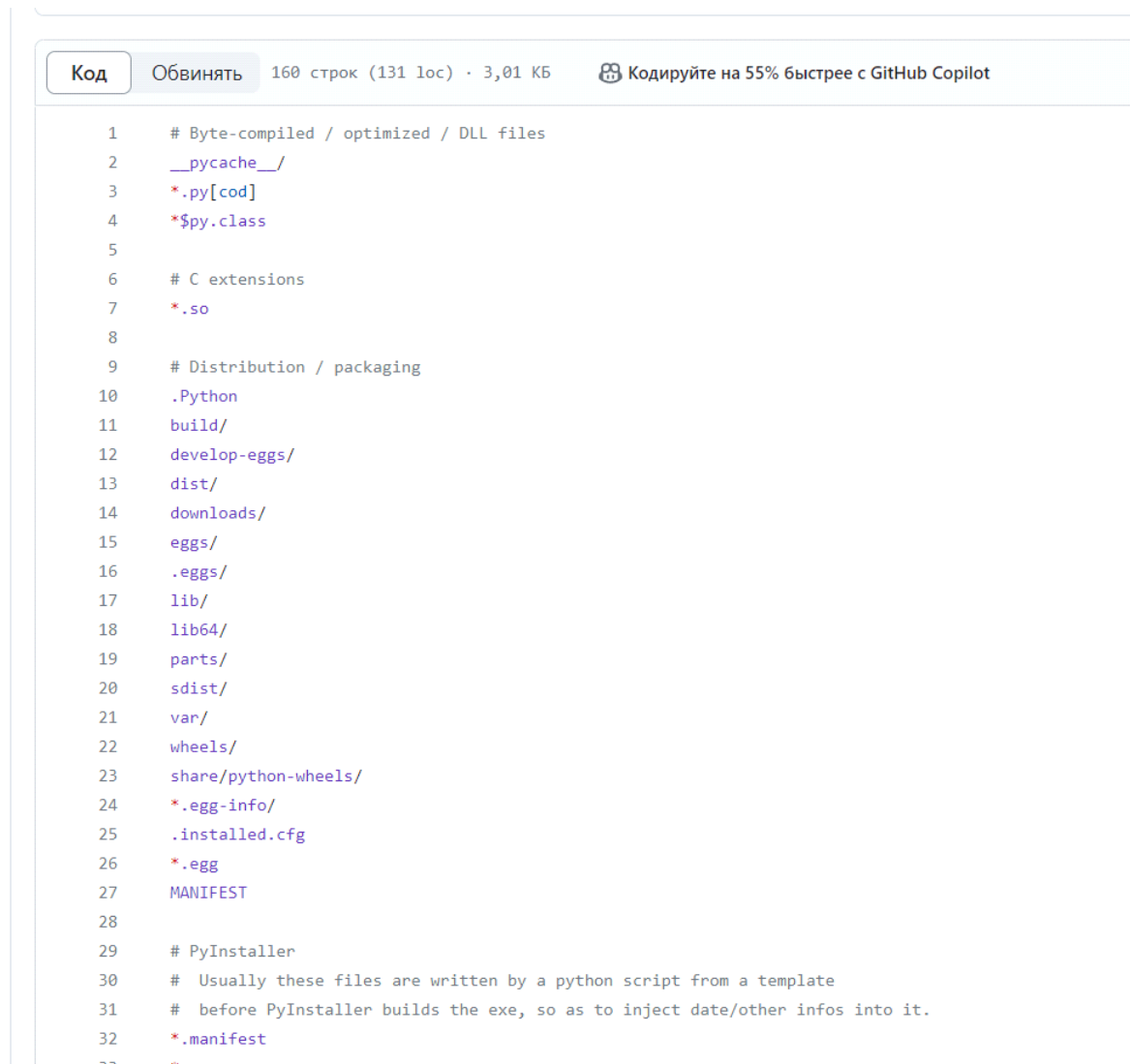
C:\git1>git clone https://github.com/mryadskaya/lab.git
Cloning into 'lab'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
Receiving objects: 100% (5/5), done./4)
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0

C:\git1>

```

Рисунок 3.1 – Копирование ссылки репозитория

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

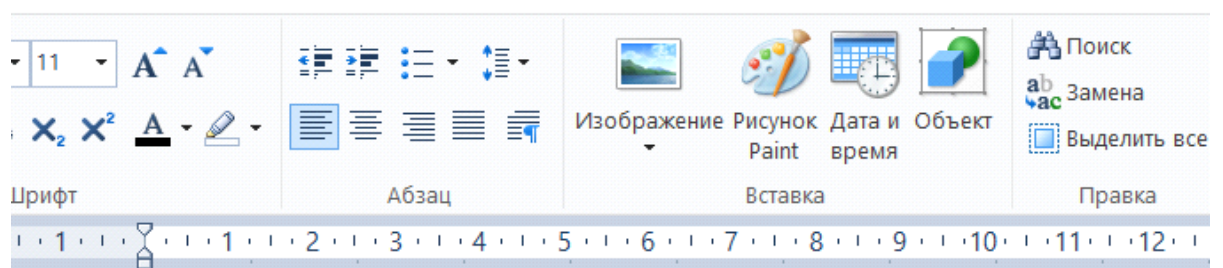
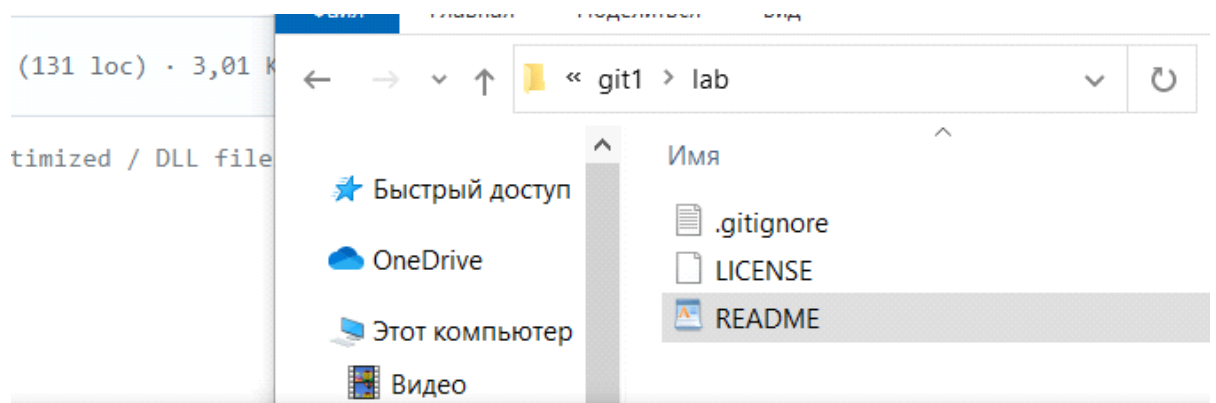


```
Код  Обвинять  160 строк (131 loc) · 3,01 КБ  Кодируйте на 55% быстрее с GitHub Copilot

1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.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 .eggs/
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 * __pycache__
```

Рисунок 41 – Файл с необходимыми правилами для языка Python

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



```
# lab
Рядская Мария Александровна
ПИЖ-б-о-22-1
```

Рисунок 5.1 – Локально создаю файл README.md

```
no changes added to commit (use "git add" and/or "git commit -a")

C:\git1\lab>git add .

C:\git1\lab>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

C:\git1\lab>git commit -m"add README.md file"
[main 026b434] add README.md file
1 file changed, 3 insertions(+), 1 deletion(-)

C:\git1\lab>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 326 bytes | 163.00 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/lab.git
   7ad0c13..026b434  main -> main

C:\git1\lab>
```

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

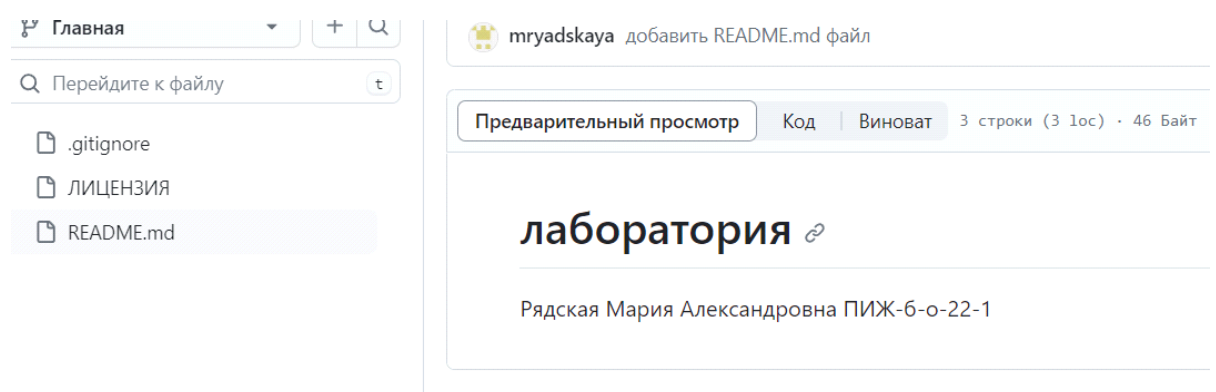


Рисунок 5.3 – Мой файл README.md на GitHub

- 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

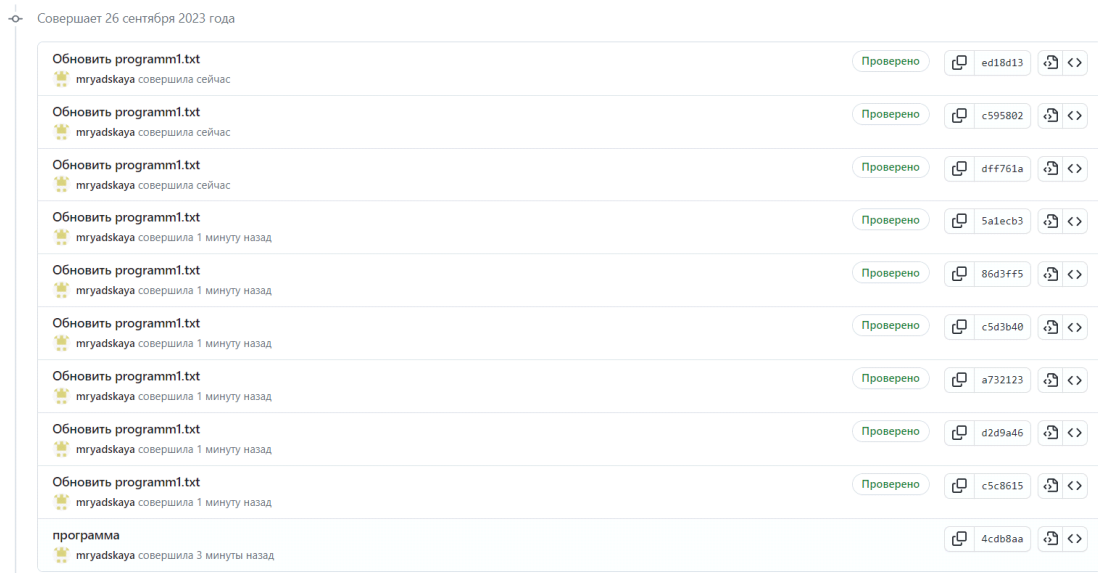


Рисунок 6 – Изменения, отображённые на GitHub

```
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

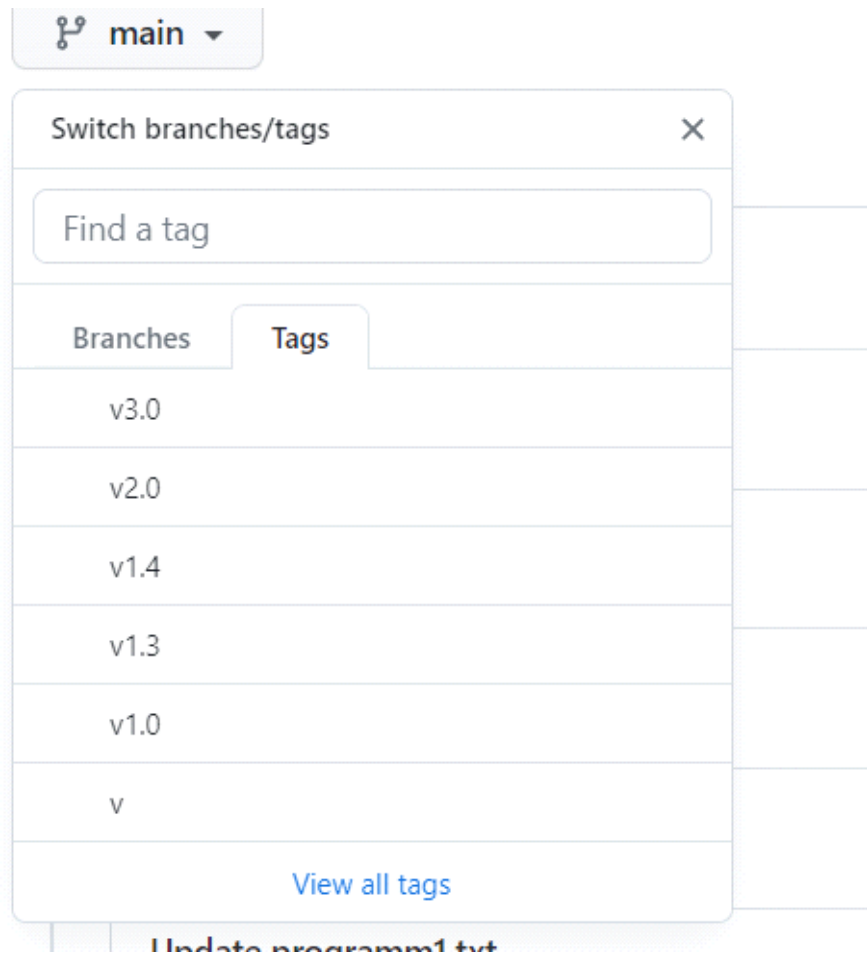


Рисунок 7.1 – Отмеченные тэги, отображённые на GitHub

- 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) prigr
mm
* 026b434 add README.md file
* 7ad0c13 Initial commit
C:\git1\lab>
```

Рисунок 8 – История хранилища

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

```

C:\git1\lab>git show HEAD
commit 4cdb8aae127b422757483aaf6aa5566ebc00d079 (HEAD -> main, tag: v3.0, tag: v2.0, tag: v1.4, tag: v1.3, tag: v1.0, tag: v, origin/main, origin/HEAD)
Author: mryadskaya <125180218+mryadskaya@users.noreply.github.com>
Date: Tue Sep 26 20:20:20 2023 +0300

    prigramm

diff --git a/programm1.txt b/programm1.txt
new file mode 100644
index 0000000..ff3fafc
--- /dev/null
+++ b/programm1.txt
@@ -0,0 +1,7 @@
+var_1 = 15
+var_2 = 67
+print('var_1:', var_1, 'var_2:', var_2)
+var_temp = var_1
+var_1 = var_2
+var_2 = var_temp
+print('var_1:', var_1, 'var_2:', var_2)
\ No newline at end of file
C:\git1\lab>

```

Рисунок 9 – Вывод последнего коммита с помощью команды
gitshowHEAD

```

C:\git1\lab>git show HEAD~1
commit 026b4340b8d47cb82b1d66c54f65c18e21dd5793
Author: mryadskaya <125180218+mryadskaya@users.noreply.github.com>
Date: Tue Sep 26 20:12:48 2023 +0300

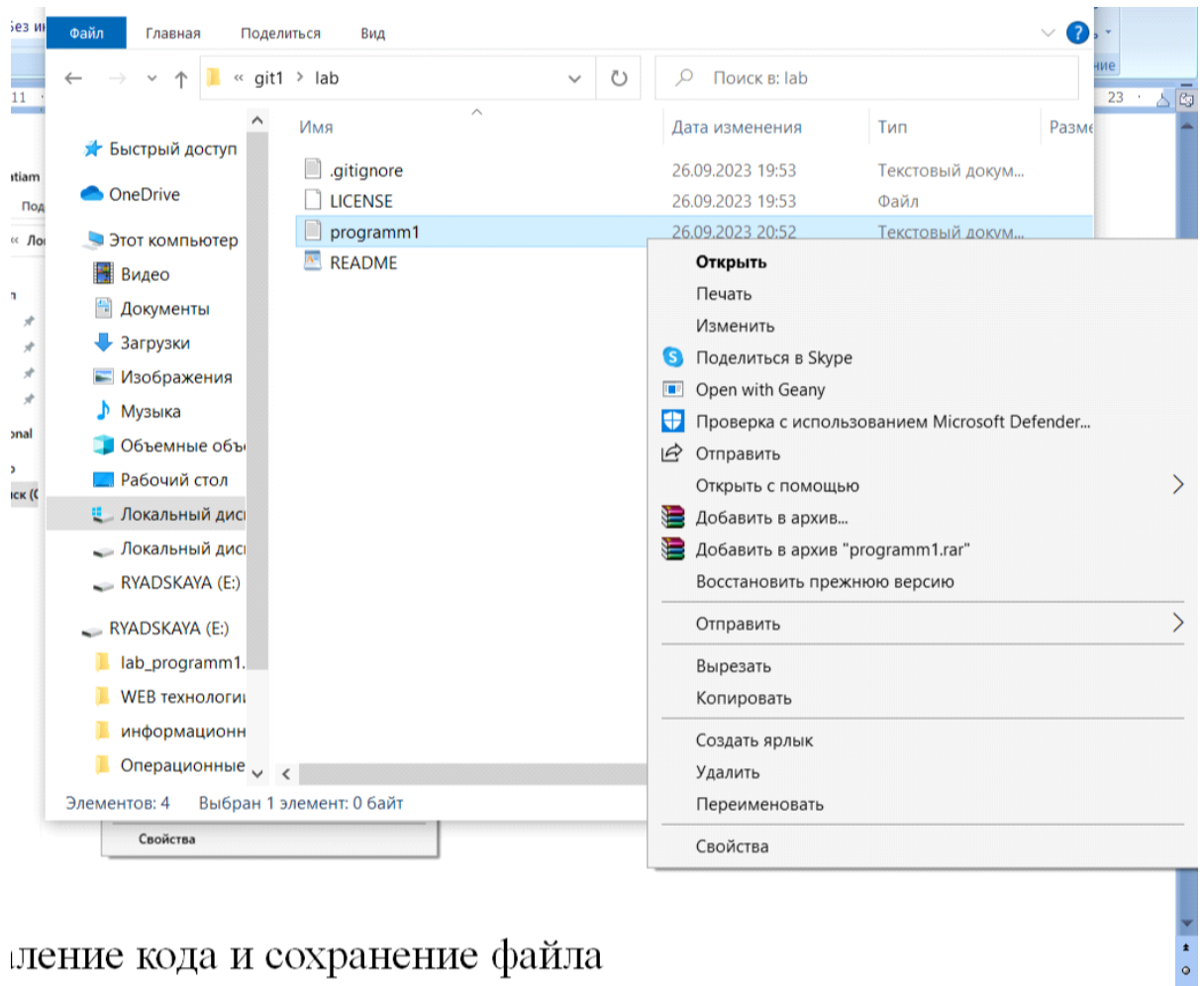
    add README.md file

diff --git a/README.md b/README.md
index 107e9f2..c7afa64 100644
--- a/README.md
+++ b/README.md
@@ -1,3 @@
-# lab
\ No newline at end of file
+# lab
+<D0><FF><E4><F1><EA><E0><FF> <CC><E0><F0><E8><FF> <C0><EB><E5><EA><F1><E0><ED><E4><F0><EE><E2><ED><E0>
+<CF><C8><C6> <E1>-<EE>-22-1
\ No newline at end of file

```

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

- Я освоила возможность отката к заданной версии
- Удалила весь код из одного из файлов программы репозитория, в моём случаеIamyour.ру, и сохранила этот файл.



ление кода и сохранение файла

Рисунок 10.1– Удаление кода и сохранение файла

- Удалила все несохраненные изменения в файле командой: `git checkout -- 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 `git checkout -- Iamyour.py`

- 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

- 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

```
C:\git1\lab>git pull
Updating 4576410..8dbe317
Fast-forward
  programm1.txt | 4 ++--
1 file changed, 2 insertions(+), 2 deletions(-)

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

C:\git1\lab>git add .
C:\git1\lab>git commit -m"I return the deleted file"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\git1\lab>git pull
Already up to date.

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

C:\git1\lab>
```

Рисунок 10.4.1 – Возвращаем предпоследний коммит

Вывод:Git предоставляет 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 `git checkout -- <имя файла>` можно вернуть файл к последнему фиксированному состоянию, а с помощью команды `git reset --hard HEAD~1` откатить состояние хранилища к предыдущей версии. То есть даже после выполнения коммита можно вернуть предыдущую версию программы.

```
C:\git1\lab>git clone https://gitlab.com/pj6233939/lb2.git
Cloning into 'lb2'...
info: please complete authentication in your browser...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

```

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:eUXGGm1YGsMAS7vkcx6J0JdOGHPem5gQp4taiCfCLB8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?

```

Auto DevOps
It will automatically build, test, and deploy your application based on a predefined CI/CD configuration.
Learn more in the [Auto DevOps documentation](#)

[Enable in settings](#)

L

lab

Project ID: 50945668

18 Commits

1 Branch

6 Tags

1.9 MiB Project Storage

Unverified

8e98a25e

main

lab

+

History

Find file

Edit

Download

Clone

README

MIT License

Add CHANGELOG

Add CONTRIBUTING

Add Kubernetes cluster

Set up CI/CD

Add Wiki

Configure Integrations

Name	Last commit	Last update
doc	Add files via upload	45 minutes ago
lb2 @ 3fe37ecd	pr2	1 week ago
.gitignore	Initial commit	1 week ago
LICENSE	Initial commit	1 week ago
README.md	add README.md file	1 week ago
programm1.txt	programm1	17 hours ago

README.md

lab

---22-1

Контрольные вопросы

- 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

- 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.

- 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`

- Как отменить индексацию файла в Git?

Используйте `gitreset HEAD <имя_файла>...` для отмены индексации файла

- Как отменить изменения в файле?

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

- Что такое удаленный репозиторий Git?

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

- Как выполнить просмотр удаленных репозиториях данного локального репозитория?

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

- 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>`

- 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 `gitfetch` или `gitpull`

- Как выполнить просмотр удаленного репозитория?

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

- Каково назначение тэгов Git?

Эта функциональность используется для отметки моментов выпуска версий

- 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 - а 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`.

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

Команда `--prune` вводится 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.