

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего
образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ»**

**Кафедра
инфокоммуникаций
Институт цифрового
развития**

**ОТЧЁТ
по лабораторной работе №1.2**

Дисциплина: «Основы программной инженерии»

Тема: «Исследование возможностей Git для работы с локальными
репозиториями»

Выполнил: студент 2
курса группы Пиж-б-о-
21-1
Рязанцев Матвей
Денисович

Ставрополь 2022

Выполнение работы

1. Был создан общедоступный репозиторий lab_1.2 на GitHub в котором будет использована лицензия MIT и язык программирования C++

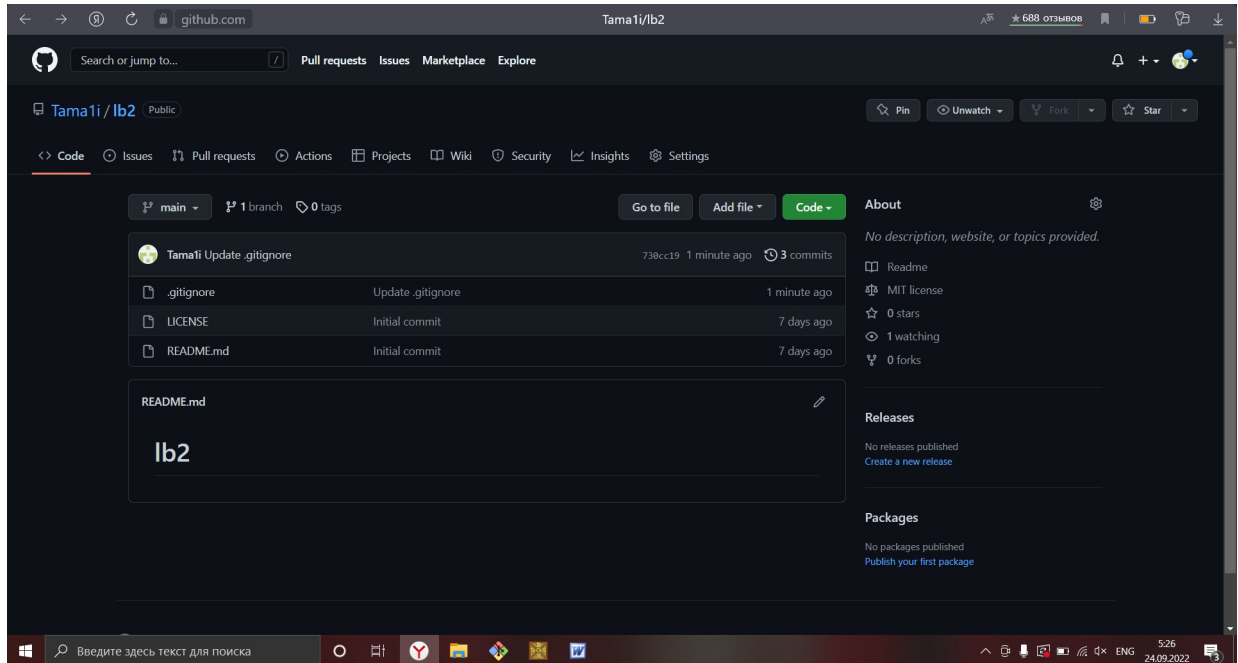


Рисунок 1 – Созданный репозиторий

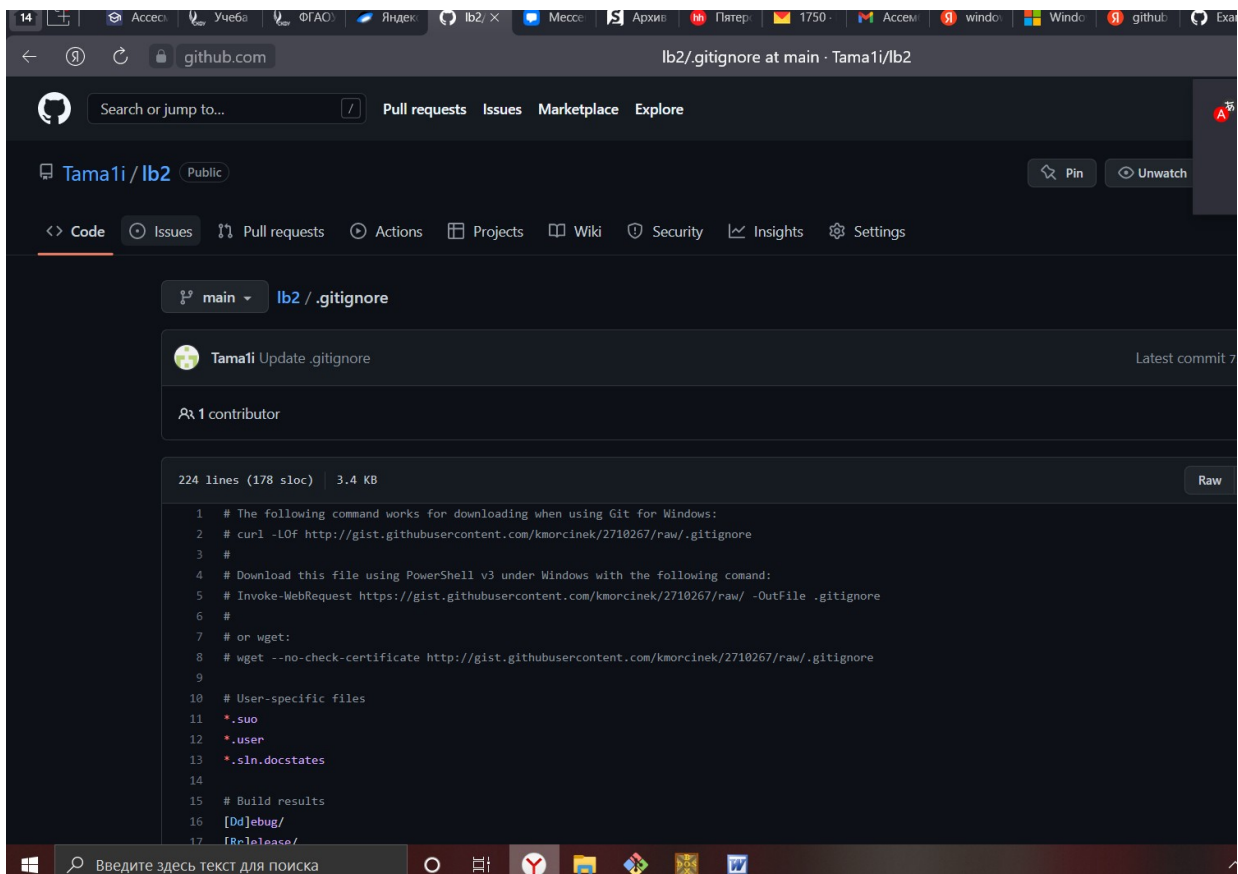


Рисунок 2 – Изменения в файле gitignore для выбранного мной языка программирования

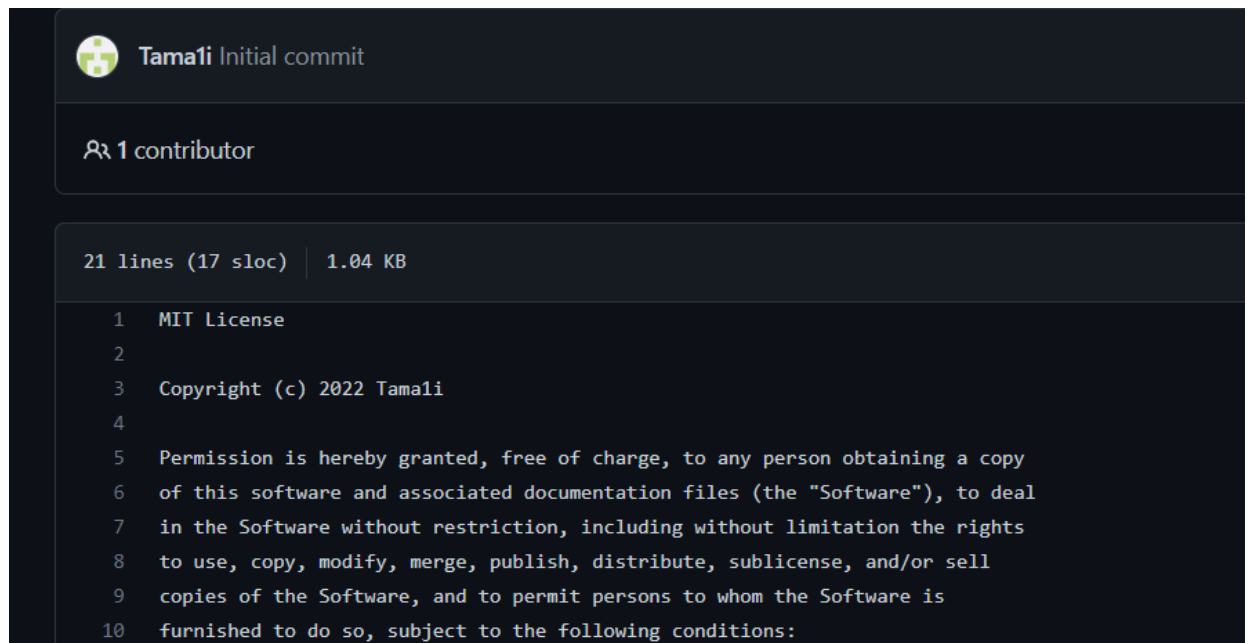


Рисунок 3 – Выбранная лицензия MIT

2. Был клонирован репозиторий на рабочий компьютер



Рисунок 4 – Клонирование репозитория

3. Добавление информации в README

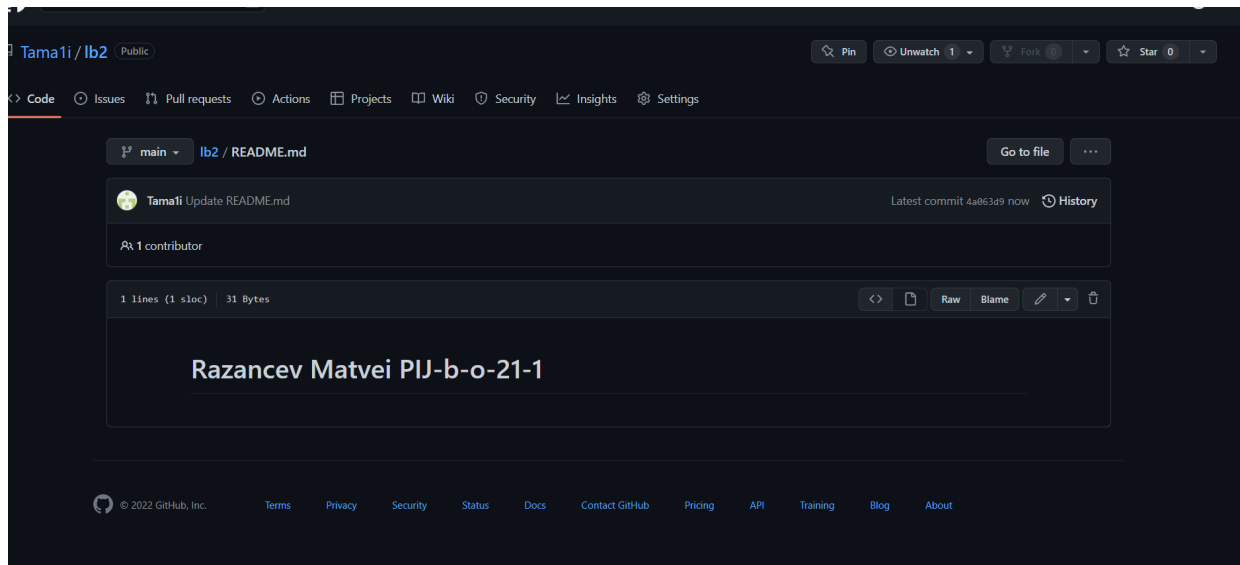


Рисунок 5 – Информация о себе в файле README

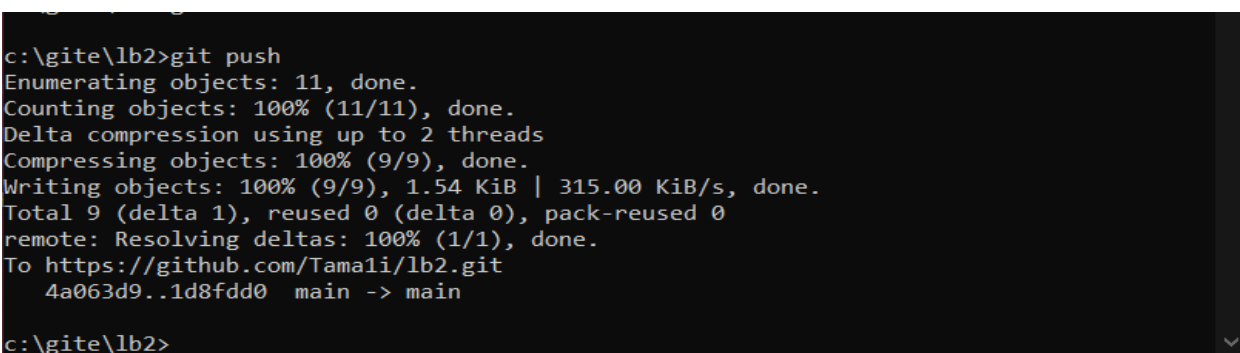


Рисунок 6 – Коммит файла README

4. Было сделано 8 коммитов и добавлено 3 тега

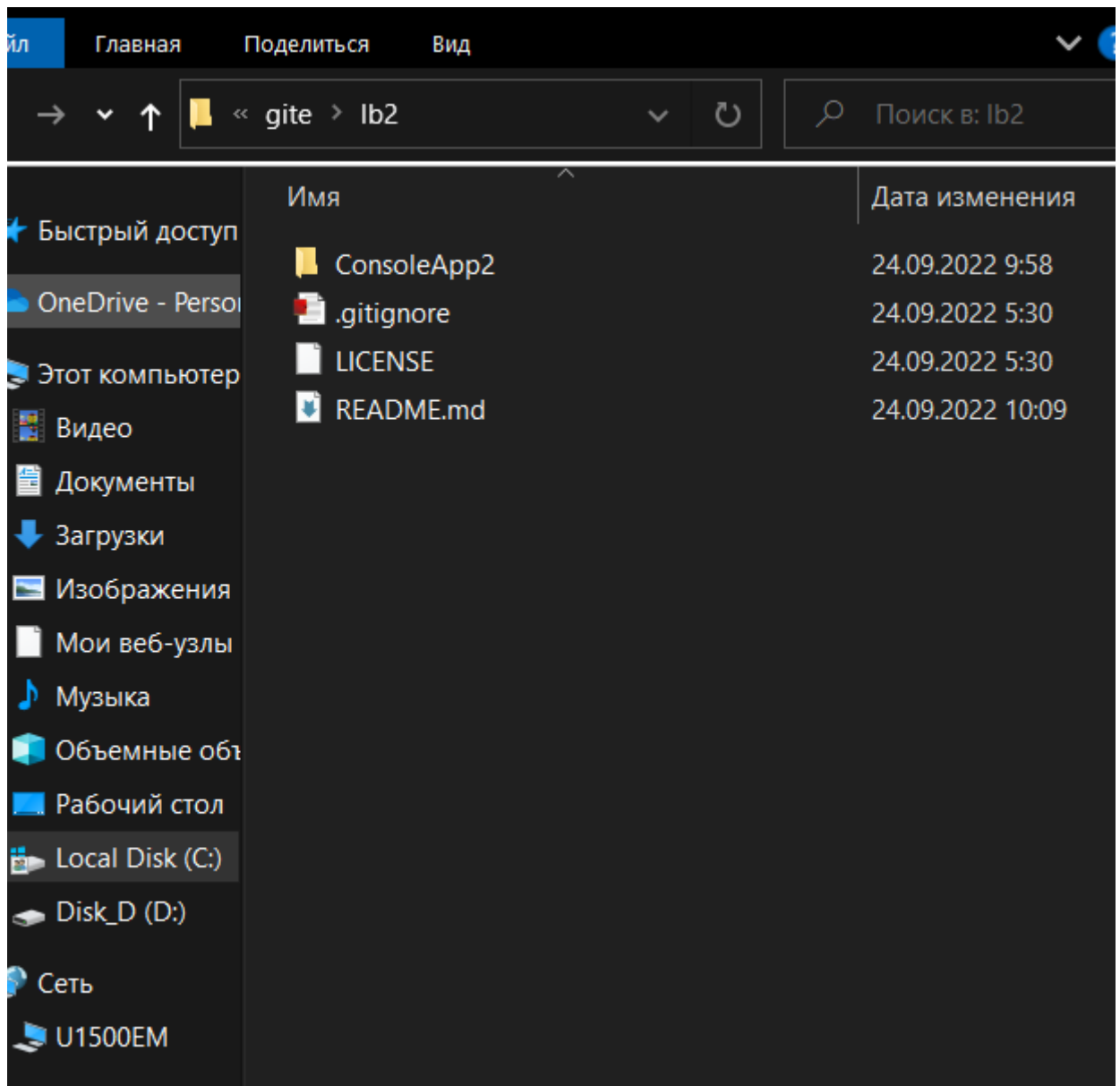


Рисунок 7 – Была создана папка для хранения проекта

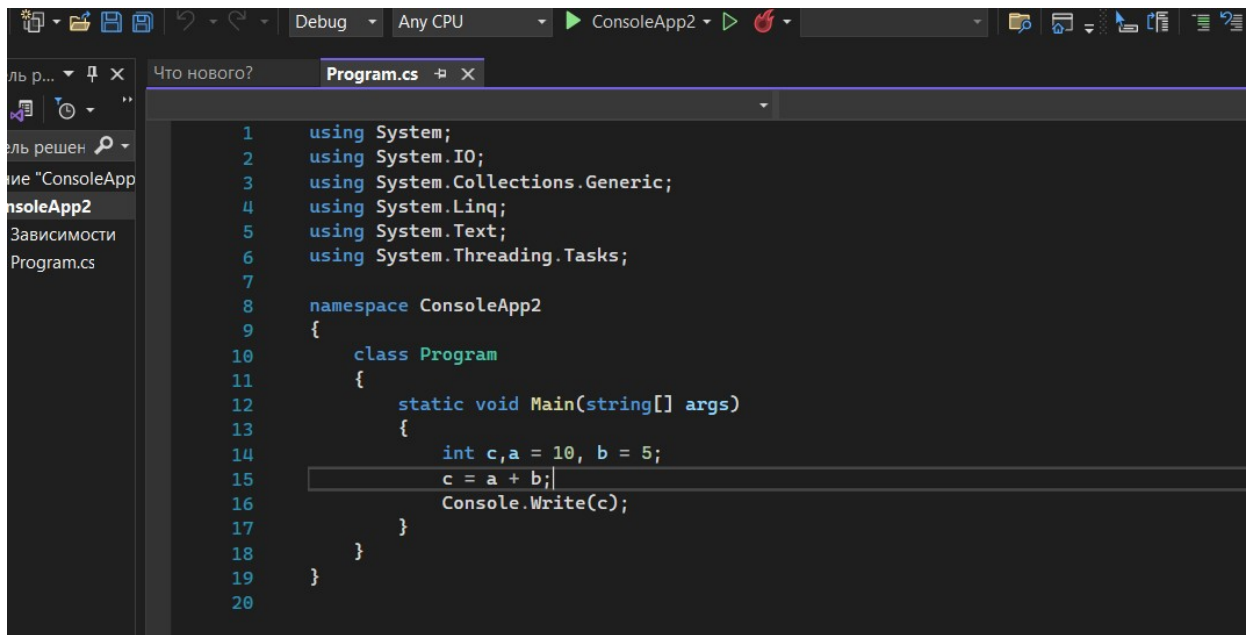


Рисунок 8 – Код программы

```
c:\gite\lb2>git add .
c:\gite\lb2>git commit -m "2com"
[main 0be4535] 2com
1 file changed, 2 insertions(+), 1 deletion(-)
c:\gite\lb2>git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 473 bytes | 118.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Tamali/lb2.git
1d8fdd0..0be4535 main -> main
```

Рисунок 9 – Коммит и пуш программы на удаленный сервис

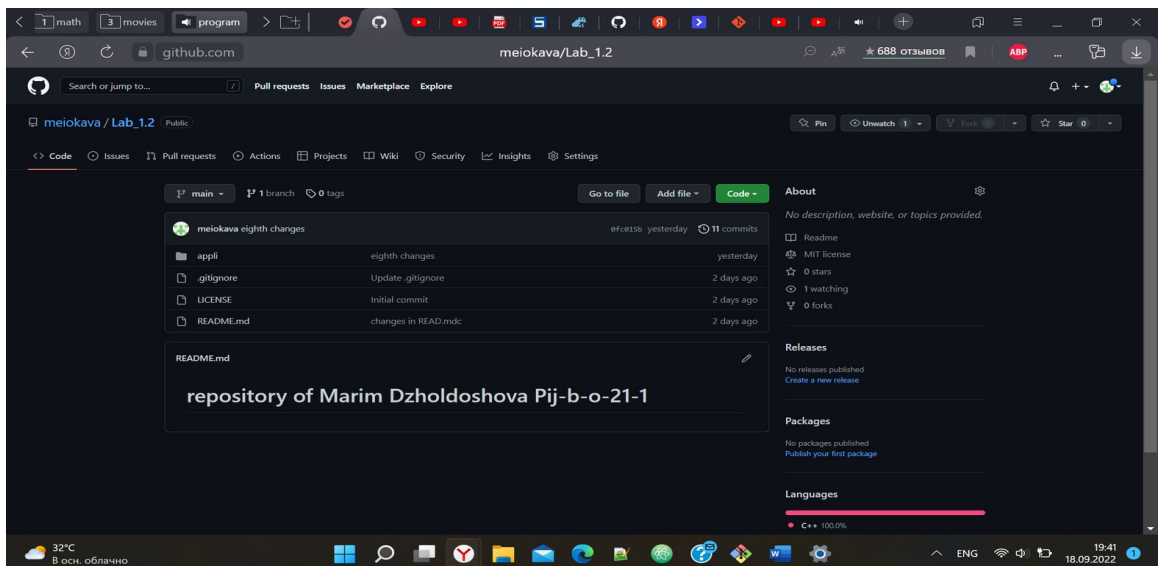


Рисунок 10 – Изменения на удаленном сервере

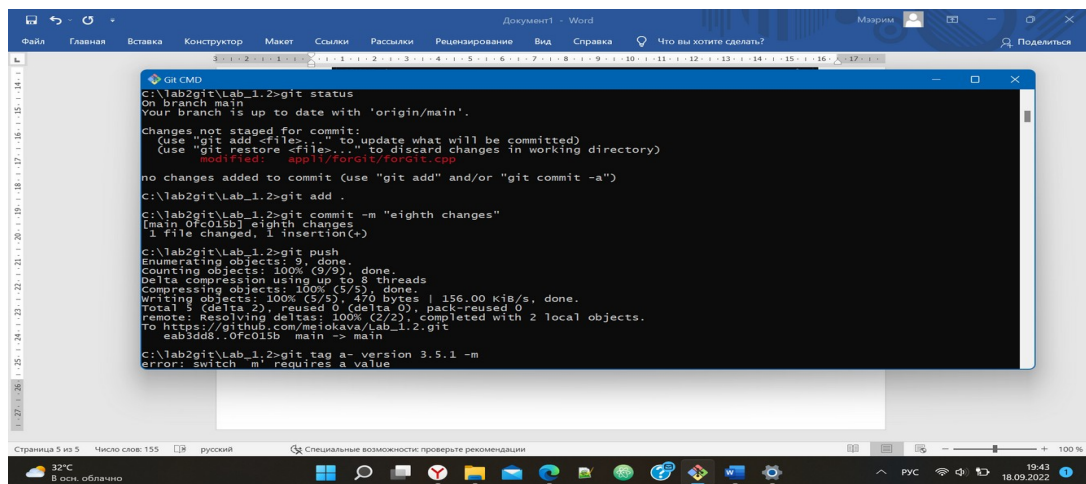


Рисунок 11 – Было сделано 8 коммитов

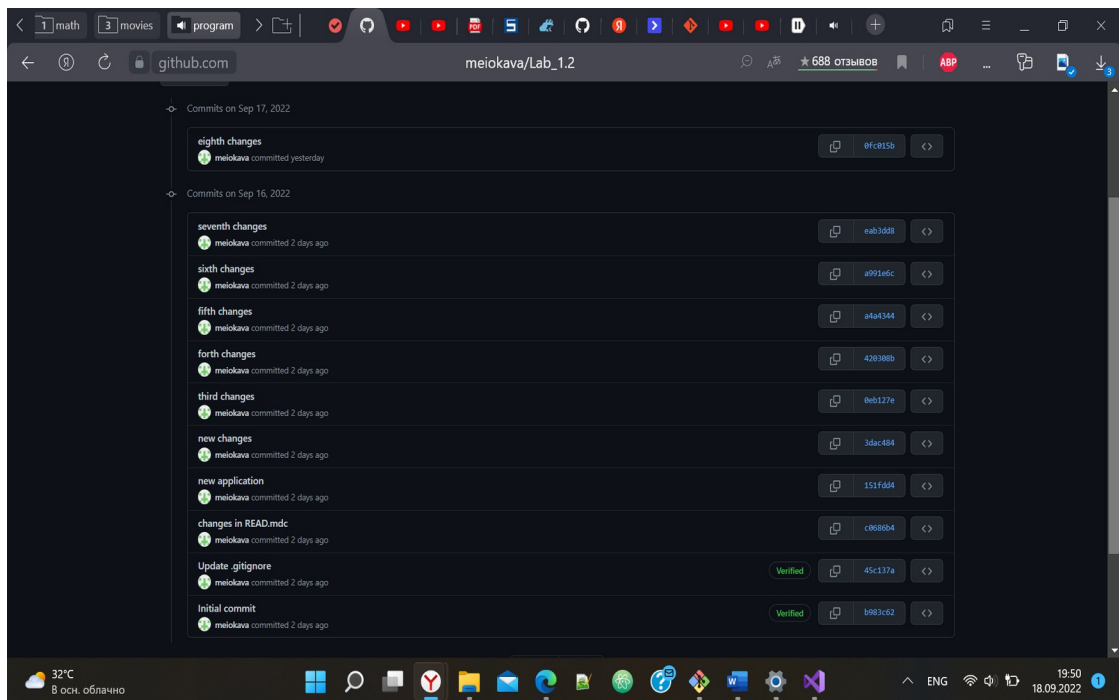


Рисунок 12 – История коммитов на удаленном сервере

```

git-cmd.exe — ярлык

c:\gite\lb2>git tag -a ver-2.5 -m "beta version 2.5"

c:\gite\lb2>git tag
ver-2.5

c:\gite\lb2>git show
commit 0be453553d0e4a3da624aaf624b23419e465f309 (HEAD -> main, tag: ver-2.5, origin/main, origin/HEAD)
Author: Tama1i <ryazantseff.matvei@yandex.ru>
Date: Sat Sep 24 10:19:07 2022 +0300

    2com

diff --git a/ConsoleApp2/ConsoleApp2/Program.cs b/ConsoleApp2/ConsoleApp2/Program.cs
index 95a6481..d7442c2 100644
--- a/ConsoleApp2/ConsoleApp2/Program.cs
+++ b/ConsoleApp2/ConsoleApp2/Program.cs
@@ -12,7 +12,8 @@ namespace ConsoleApp2
     static void Main(string[] args)
     {
         int c,a = 10, b = 5;
-        c = a + b;
+        c = a + b+1;
+        c += 1;
         Console.Write(c);
     }
 }

c:\gite\lb2>

```

Рисунок 13 – Создание аннотированного тега


```

c:\gite\lb2>git add .

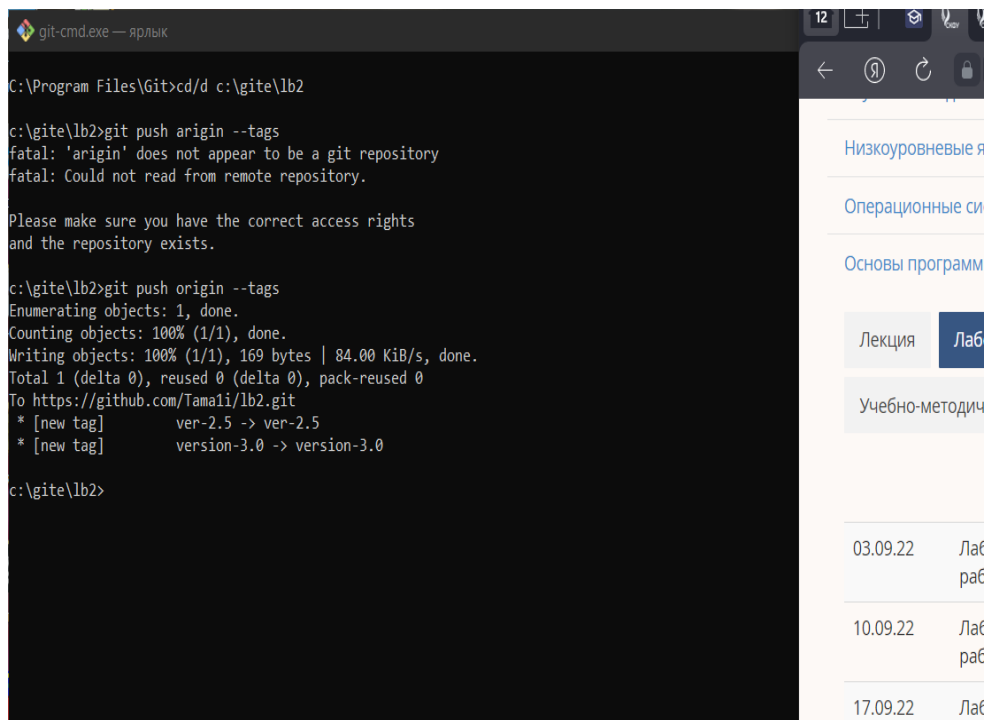
c:\gite\lb2>git commit -m "3com"
[main aba6138] 3com
 1 file changed, 1 insertion(+), 1 deletion(-)

c:\gite\lb2>git tag version-3.0

c:\gite\lb2>git tag -n
ver-2.5          beta version 2.5
version-3.0      3com

```

Рисунок 14 – Создание легковесного тега



```

C:\Program Files\Git>cd/d c:\gite\lb2

c:\gite\lb2>git push arigin --tags
fatal: 'arigin' does not appear to be a git repository
fatal: Could not read from remote repository.

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

c:\gite\lb2>git push origin --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 169 bytes | 84.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Tamali/lb2.git
 * [new tag]         ver-2.5 -> ver-2.5
 * [new tag]         version-3.0 -> version-3.0

c:\gite\lb2>

```

Рисунок 16 – Отправка тегов на удаленный сервер

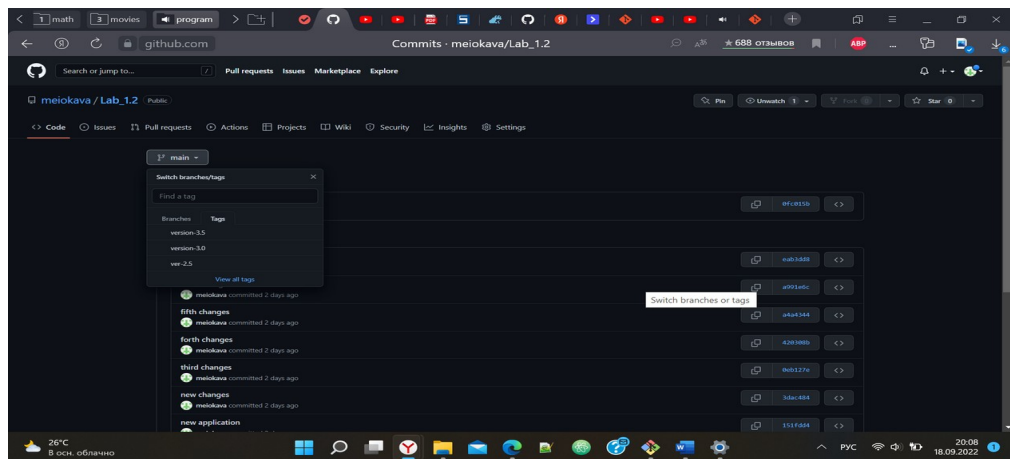


Рисунок 17 – История тегов на удаленном сервере

```

git-cmd.exe — ярлык - "C:\Program Files\Git\cmd\git.exe" log
commit 7b89192797788ada4bf3801749efda63193e415f (HEAD -> main, origin/main, origin/H ^
EAD)
Author: Tama1i <ryazantseff.matvei@yandex.ru>
Date: Sat Sep 24 10:43:34 2022 +0300

    8com

commit bf40f01c5da4a006334a63b08d27e2dfc0f760e3
Author: Tama1i <ryazantseff.matvei@yandex.ru>
Date: Sat Sep 24 10:41:37 2022 +0300

    6 com'

commit dab840697afc31e7faab330c2d9f54c93a012c6a
Author: Tama1i <ryazantseff.matvei@yandex.ru>
Date: Sat Sep 24 10:39:42 2022 +0300

    5 com

commit 6c6935f85429553a13bd3afcc000ec5b1b028d31
Author: Tama1i <ryazantseff.matvei@yandex.ru>
Date: Sat Sep 24 10:31:44 2022 +0300

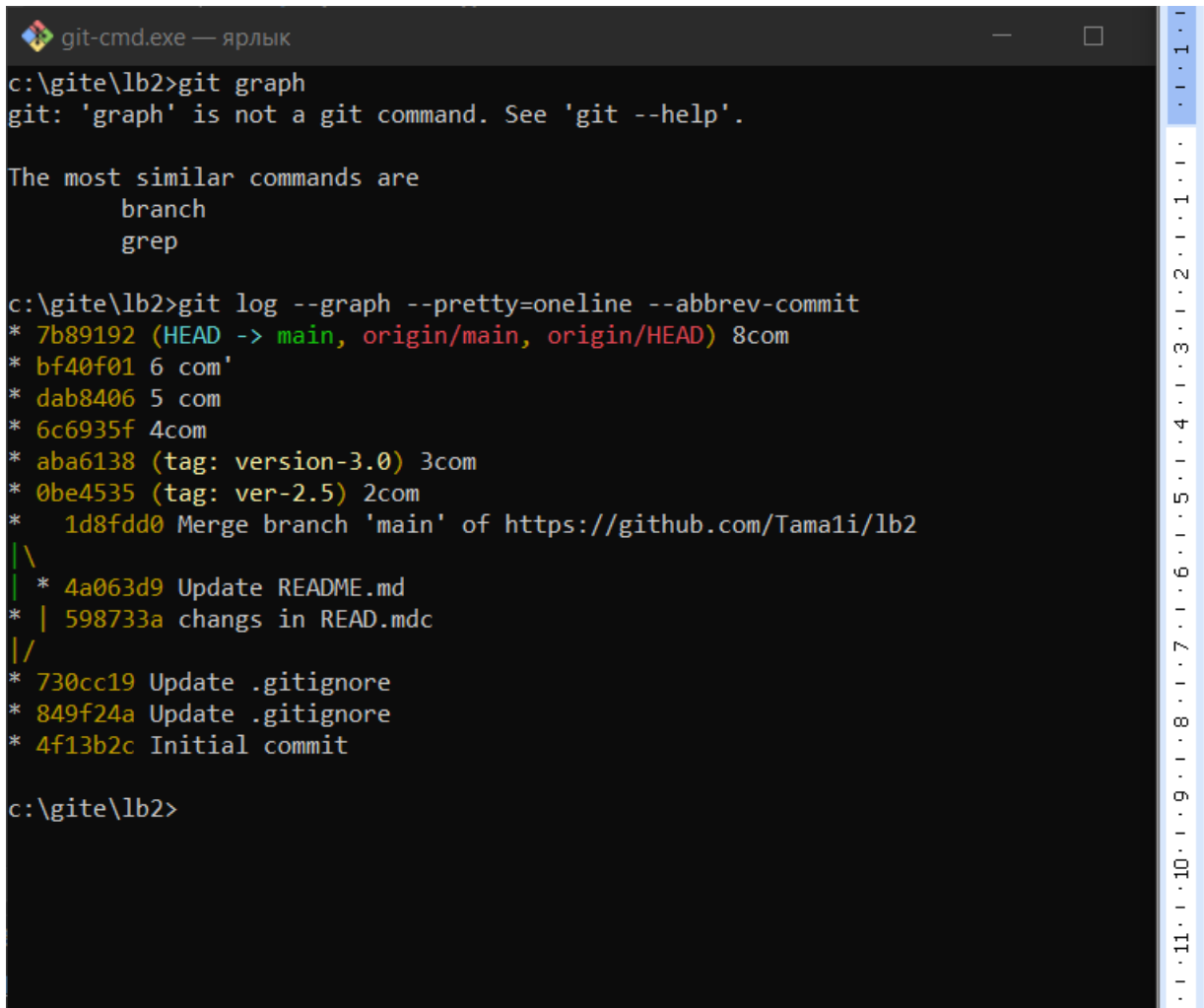
    4com

commit aba6138deef33b1ab509b28a0260b5954e6520f9 (tag: version-3.0)
Author: Tama1i <ryazantseff.matvei@yandex.ru>
Date: Sat Sep 24 10:29:15 2022 +0300

:

```

Рисунок 18 – История коммитов при помощи команды git log



```
c:\gite\lb2>git graph
git: 'graph' is not a git command. See 'git --help'.

The most similar commands are
    branch
    grep

c:\gite\lb2>git log --graph --pretty=oneline --abbrev-commit
* 7b89192 (HEAD -> main, origin/main, origin/HEAD) 8com
* bf40f01 6 com'
* dab8406 5 com
* 6c6935f 4com
* aba6138 (tag: version-3.0) 3com
* 0be4535 (tag: ver-2.5) 2com
* 1d8fdd0 Merge branch 'main' of https://github.com/Tama1i/lb2
| \
| * 4a063d9 Update README.md
| * | 598733a changs in README.mdc
| /
* 730cc19 Update .gitignore
* 849f24a Update .gitignore
* 4f13b2c Initial commit

c:\gite\lb2>
```

Рисунок 20 – Просмотр коммитов командой git graph

5. Просмотрел содержимое коммитов командой git show HEAD, git show HEAD~1, git show a991e6c:

```
git-cmd.exe — ярлык
| * 4a063d9 Update README.md
| * | 598733a changes in README.mdc
| /
| * 730cc19 Update .gitignore
| * 849f24a Update .gitignore
| * 4f13b2c Initial commit

c:\gite\lb2>git show head
commit 7b89192797788ada4bf3801749efda63193e415f (HEAD -> main, origin/main, origin/H
EAD)
Author: Tamali <ryazantseff.matvei@yandex.ru>
Date: Sat Sep 24 10:43:34 2022 +0300

    8com

diff --git a/ConsoleApp2/ConsoleApp2/Program.cs b/ConsoleApp2/ConsoleApp2/Program.cs
index a5bee53..4aa669b 100644
--- a/ConsoleApp2/ConsoleApp2/Program.cs
+++ b/ConsoleApp2/ConsoleApp2/Program.cs
@@ -13,7 +13,7 @@ namespace ConsoleApp2
    {
        int c,a = 10, b = 5;
        c = a + b+1;
-       c += 5;
+       c += 6;
        Console.Write(c);
    }
}

c:\gite\lb2>
```

Рисунок 21 – Просмотр содержимого последнего коммита

```
git-cmd.exe — ярлык

        int c,a = 10, b = 5;
        c = a + b+1;
-       c += 5;
+       c += 6;
        Console.Write(c);
    }
}

c:\gite\lb2>git show head~1
commit bf40f01c5da4a006334a63b08d27e2dfc0f760e3
Author: Tamali <ryazantseff.matvei@yandex.ru>
Date: Sat Sep 24 10:41:37 2022 +0300

    6 com'

diff --git a/ConsoleApp2/ConsoleApp2/Program.cs b/ConsoleApp2/ConsoleApp2/Program.cs
index d06a6d8..a5bee53 100644
--- a/ConsoleApp2/ConsoleApp2/Program.cs
+++ b/ConsoleApp2/ConsoleApp2/Program.cs
@@ -13,7 +13,7 @@ namespace ConsoleApp2
    {
        int c,a = 10, b = 5;
        c = a + b+1;
-       c += 4;
+       c += 5;
        Console.Write(c);
    }
}

c:\gite\lb2>
```

Рисунок 22 – Просмотр предпоследнего коммита

```
git-cmd.exe — ярлык

    int c,a = 10, b = 5;
    c = a + b+1;
-   c += 4;
+   c += 5;
    Console.Write(c);
}
}

c:\gite\lb2>git show bf40f01c
commit bf40f01c5da4a006334a63b08d27e2dfc0f760e3
Author: Tama1i <ryazantseff.matvei@yandex.ru>
Date:   Sat Sep 24 10:41:37 2022 +0300

    6 com'

diff --git a/ConsoleApp2/ConsoleApp2/Program.cs b/ConsoleApp2/ConsoleApp2/Program.cs
index d06a6d8..a5bee53 100644
--- a/ConsoleApp2/ConsoleApp2/Program.cs
+++ b/ConsoleApp2/ConsoleApp2/Program.cs
@@ -13,7 +13,7 @@ namespace ConsoleApp2
    {
        int c,a = 10, b = 5;
        c = a + b+1;
-       c += 4;
+       c += 5;
        Console.Write(c);
    }
}

c:\gite\lb2>
```

Рисунок 23 – Просмотр коммита с указанным хэшем

6. Возможность отката к заданной версии

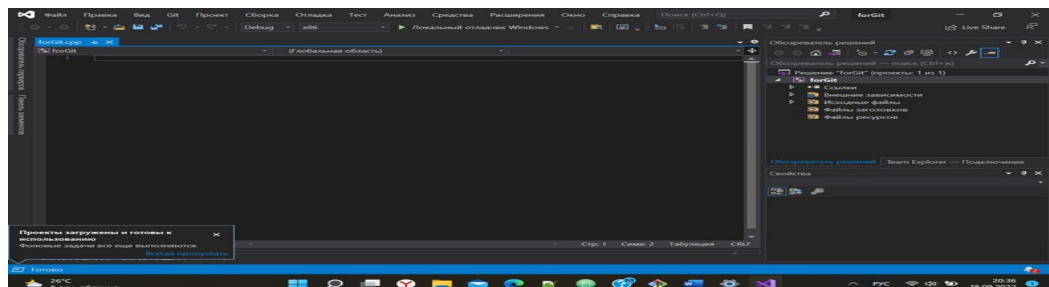


Рисунок 24 – Весь код был удален, а изменения сохранены

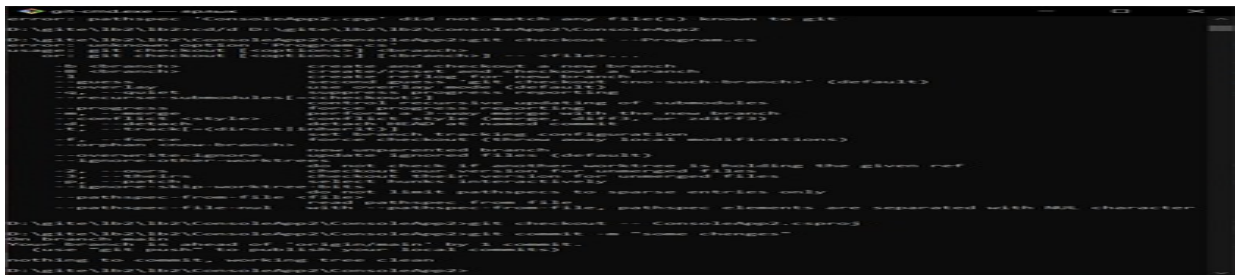


Рисунок 25 – Удаление всех несохраненных изменений командой checkout

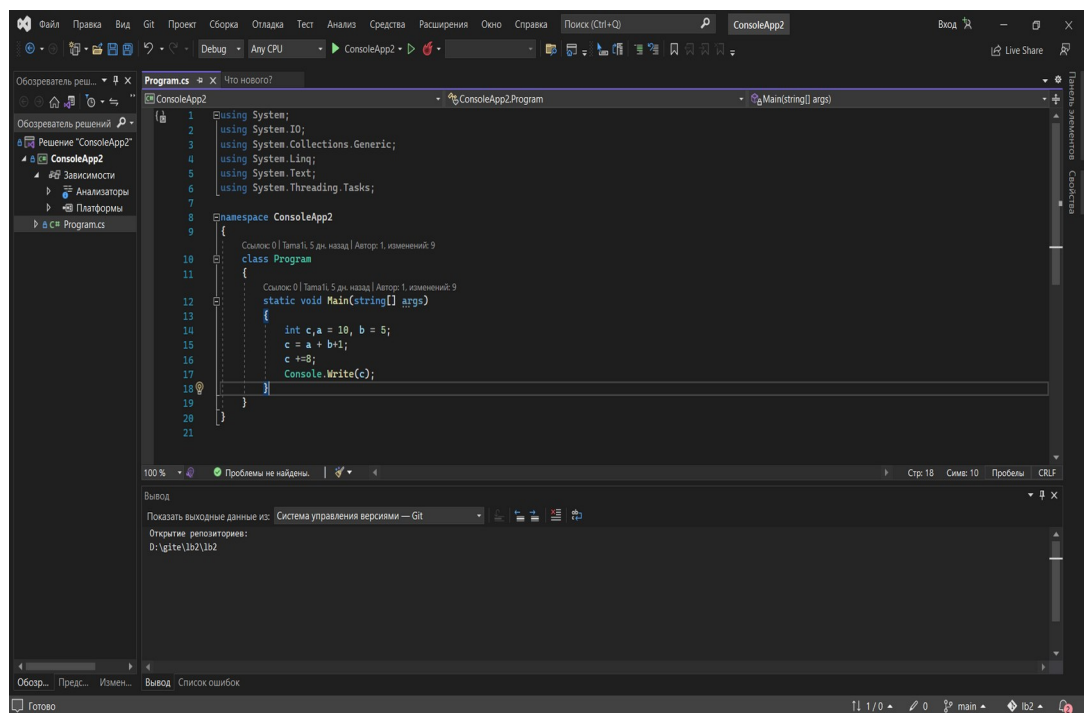


Рисунок 26 – Изменения программы после команды checkout

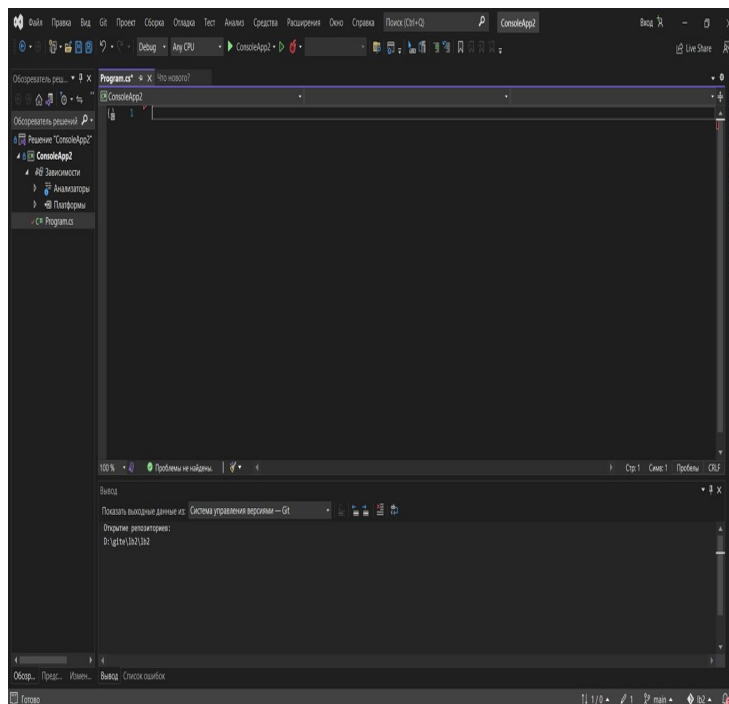


Рисунок 27 – Удаление кода в Visual studio

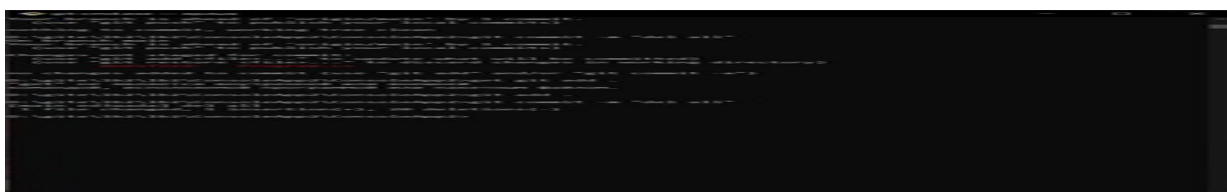


Рисунок 28 – Коммит изменений

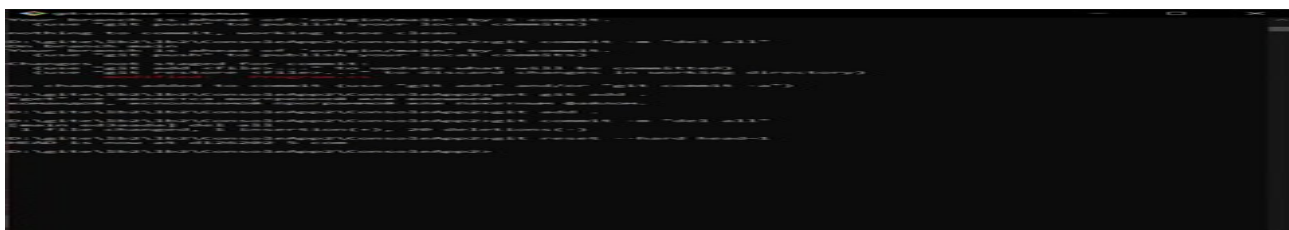


Рисунок 29 – Откат состояния хранилища к предыдущей версии

Контрольные вопросы и ответы на них:

Вопросы для защиты работы.

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

Наиболее простой и в то же время мощный инструмент для этого — команда `git log`. По умолчанию, без аргументов, `git log` выводит список

коммитов созданных в данной репозитории в обратном хронологическом порядке. То есть самые последние коммиты показываются первыми. Одна из опций, когда вы хотите увидеть сокращенную статистику для каждого коммита, вы можете использовать опцию `--stat`.

Вторая опция (одна из самых полезных аргументов) является `-p` или `--patch`, который показывает разницу (выводит патч), внесенную в каждый коммит. Так же вы можете ограничить количество записей в выводе команды; используйте параметр `-2` для вывода только двух записей (пример команды `git log -p -2`).

Третья действительно полезная опция это `--pretty`. Она меняет формат вывода. Существует несколько встроенных вариантов отображения. Опция `oneline` выводит каждый коммит в одну строку, что может быть очень удобным если вы просматриваете большое количество коммитов. К тому же, опции `short`, `full` и `fuller` делают вывод приблизительно в том же формате, но с меньшим или большим количеством информации соответственно. Наиболее интересной опцией является `format`, которая позволяет указать формат для вывода информации. Особенно это может быть полезным, когда вы хотите сгенерировать вывод для автоматического анализа — так как вы указываете формат явно, он не будет изменен даже после обновления Git. Для опции `git log --pretty=format` существуют различного рода опции для изменения формата отображения.

2. Как ограничить вывод при просмотре истории коммитов?

Для ограничения может использоваться функция `git log <n>`, где `n` число

записей. Также, существуют опции для ограничения вывода по времени, такие как `--since` и `--until`, они являются очень удобными. Например, следующая команда покажет список коммитов, сделанных за последние две недели: `git log --since=2. weeks` Это команда работает с большим количеством форматов — вы можете указать определенную дату вида `2008-01-15` или же относительную дату, например `2 years 1 day 3 minutes ago`.

Также вы можете фильтровать список коммитов по заданным параметрам. Опция `--author` дает возможность фильтровать по автору коммита, а опция `--grep` (показывает только коммиты, сообщение которых содержит указанную строку) искать по ключевым словам в сообщении коммита.

Функция `-S` показывает только коммиты, в которых изменение в коде повлекло за собой добавление или удаление указанной строки.

3. Как внести изменения в уже сделанный коммит? Внести изменения можно с помощью команды `git commit --amend`. Эта команда берёт индекс и применяет его к последнему коммиту. Если после последнего коммита не было никаких проиндексированных изменений (например, вы запустили приведённую команду сразу после предыдущего коммита), то состояние проекта будет абсолютно таким же и всё, что мы изменим, это комментарий к коммиту.

Для того, чтобы внести необходимые изменения - нам нужно проиндексировать их и выполнить команду `git commit --amend`.
`git commit -m 'initial commit'`
`git add forgotten_file`
`git commit --amend`

Эффект от выполнения этой команды такой, как будто мы не выполнили предыдущий коммит, а еще раз выполнили команду `git add` и выполнили коммит.

4. Как отменить индексацию файла в Git? Например, вы изменили два файла и хотите добавить их в разные коммиты, но случайно выполнили команду `git add *` и добавили в индекс оба. Как исключить из индекса один из них? Команда `git status` напомним вам: прямо под текстом «Changes to be committed» говорится: используйте `git reset HEAD <file>` для исключения из индекса.

5. Как отменить изменения в файле? С помощью команды `git checkout --<file>`.

6. Что такое удаленный репозиторий Git? Удалённый репозиторий — это своего рода наше облако, в которое мы сохраняем те или иные изменения

в нашей программе/коде/файлах.

7. Как выполнить просмотр удаленных репозиториях данного локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозиториях, необходимо запустить команду `git remote`. Также можно указать ключ `-v`, чтобы просмотреть адреса для чтения и записи, привязанные к репозиторию. Пример: `git remote -v`

8. Как добавить удаленный репозиторий для данного локального репозитория?

Для того, чтобы добавить удалённый репозиторий и присвоить ему имя (shortname), просто выполните команду `git remote add <shortname> <url>`.

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

Если необходимо получить изменения, которые есть у Пола, но нету у вас, вы можете выполнить команду `git fetch <Название репозитория>`. Важно отметить, что команда `git fetch` забирает данные в ваш локальный репозиторий, но не сливает их с какими-либо вашими наработками и не модифицирует то, над чем вы работаете в данный момент. Вам необходимо вручную слить эти данные с вашими, когда вы будете готовы. Если ветка настроена на отслеживание удалённой ветки, то вы можете использовать команду `git pull` чтобы автоматически получить изменения из удалённой ветки и слить их со своей текущей. Выполнение `git pull`, как правило, извлекает (fetch) данные с сервера, с которого вы изначально клонировали, и автоматически пытается слить (merge) их с кодом, над которым вы в данный момент работаете. Чтобы отправить изменения на удалённый репозиторий необходимо отправить их в удалённый репозиторий. Команда для этого действия простая: `git push <remote-name> <branch-name>`.

10. Как выполнить просмотр удаленного репозитория?

Для просмотра удалённого репозитория, можно использовать команду `git remote show <remote>`.

11. Каково назначение тэгов Git?

Теги — это ссылки, указывающие на определённые версии кода/написанной программы. Они удобны чтобы в случае чего вернуться к нужному моменту. Также при помощи тегов можно пометать важные моменты.

12. Как осуществляется работа с тэгами Git?

Просмотреть наличие тегов можно с помощью команды: `git tag`. А назначить (указать, добавить тег) можно с помощью команды `git tag -a v1.4(версия изначальная) -m "Название"`. С помощью команды `git show` вы можете посмотреть данные тега вместе с коммитом: `git show v1.4`. Отправка тегов, по умолчанию, команда `git push` не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на удалённый сервер. Процесс аналогичен отправке веток — достаточно выполнить команду `git push origin <tagname>`. Для отправки всех тегов можно использовать команду `git push origin tags`. Для удаления тега в локальной репозитории достаточно выполнить команду `git tag -d <tagname>`. Например, удалить созданный ранее легковесный тег можно следующим образом: `git tag -d v1.4-lw`. Для удаления тега из внешнего репозитория используется команда `git push origin --delete <tagname>`. Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать `git checkout` для тега пример: `git checkout -b version2 v2.0.0`.

13. Самостоятельно изучите назначение флага `--prune` в командах `git fetch` и `git push`. Каково назначение этого флага? `Git fetch --prune` команда получения всех изменений с репозитория GitHub. В команде `git push --prune` удаляет удаленные ветки, у которых нет локального аналога.