**МИНИCTEPCTBO 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 кроссплатформенного программирования»

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

|  |
| --- |
| Выполнил: студент 1 курса |
| группы ИВТ-б-о-21-1 |
| Богдашов Артём Владимирович |

Ставрополь 2022

Выполнение работы:

1. Создал общедоступный репозиторий rep\_1.2 на GitHub в котором будет использована лицензия MIT и выбранный мной язык программирования.

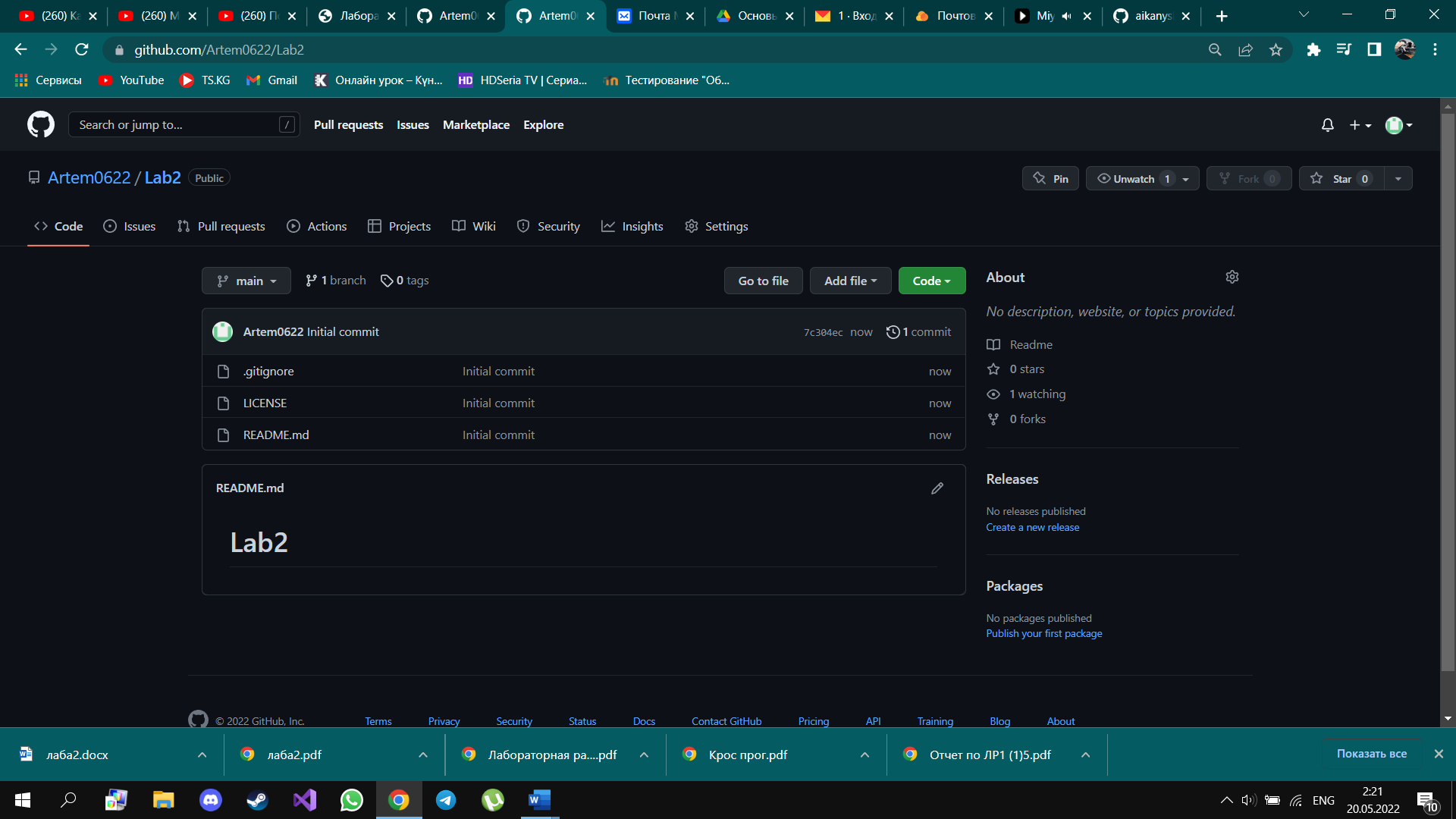


Рисунок 1.1 Созданный репозиторий в GitHub

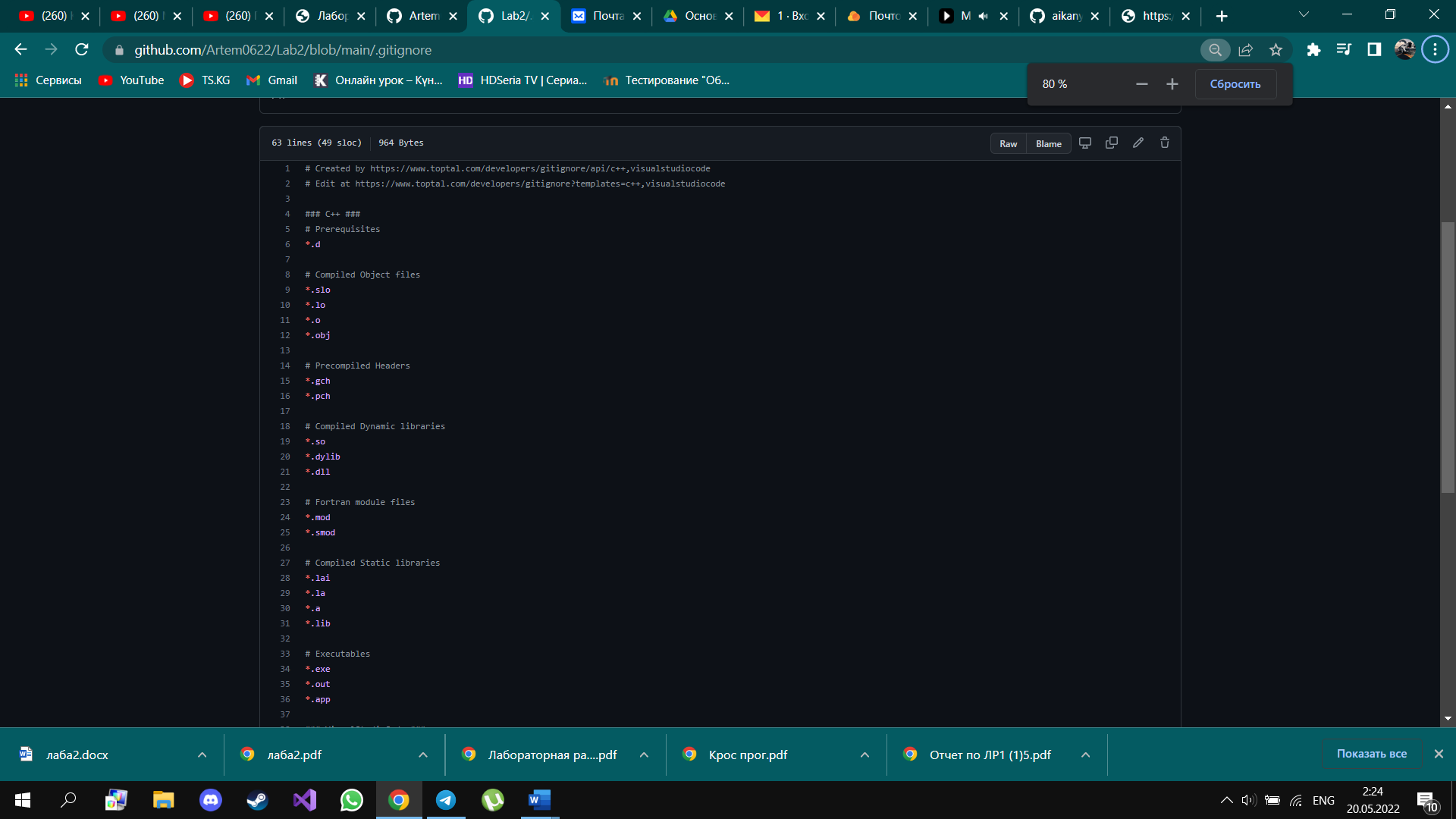


Рисунок 1.2 Изменения в файле .gitignore

2.Клонировал созданный репозиторий на раб. компьютер:

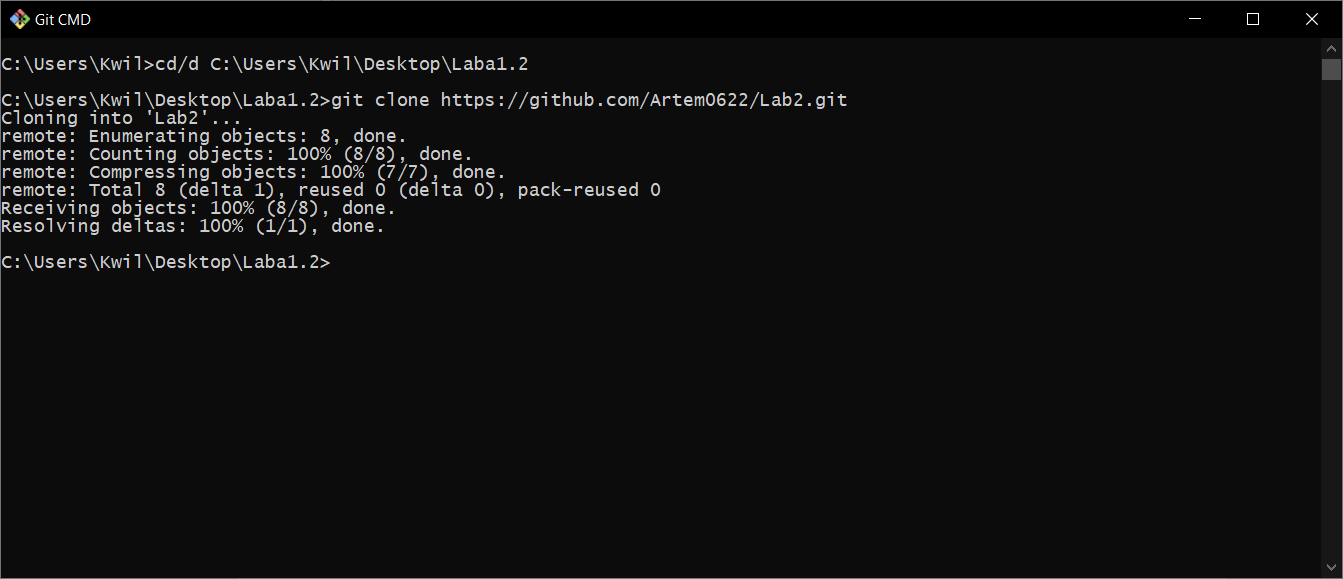


Рисунок 2. Клонирование репозитория

3.Добавил информацию в README и закоммитил:

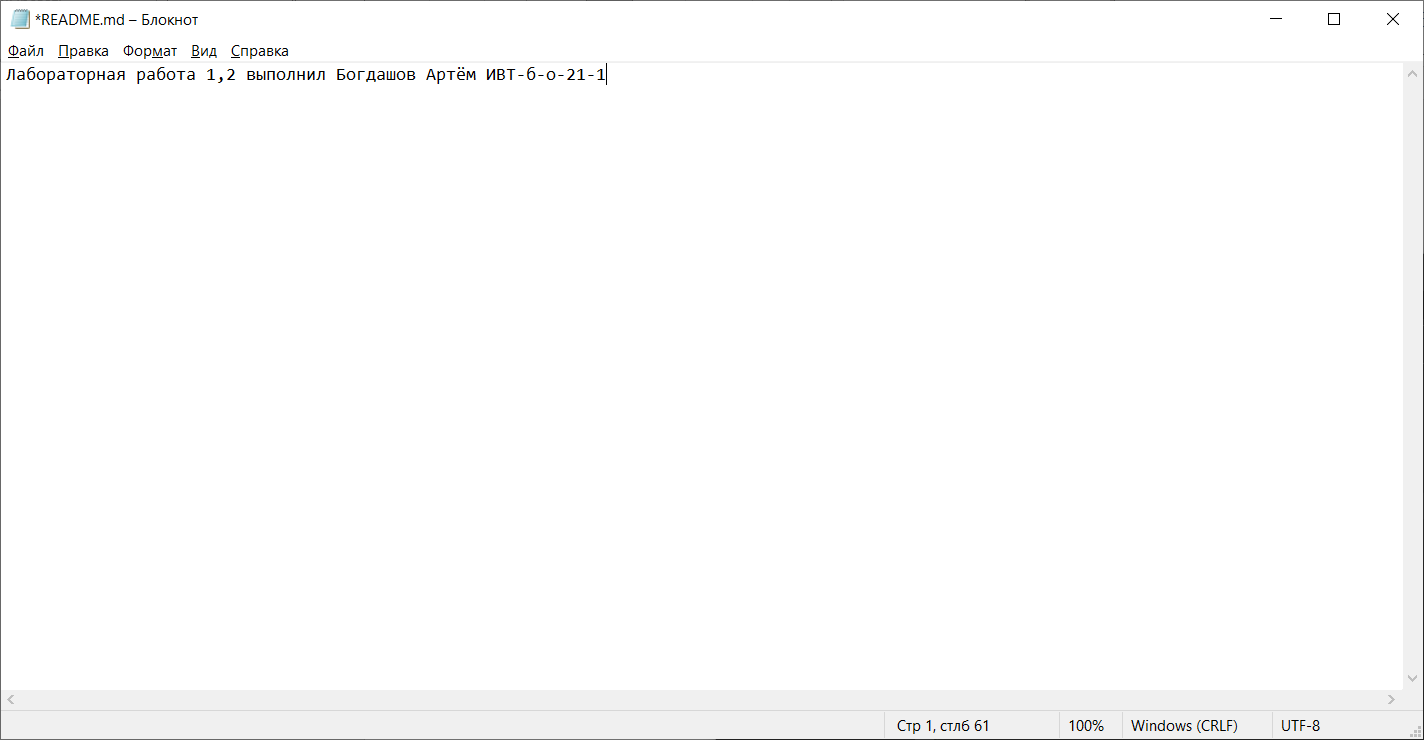


Рисунок 3.1 Добавление информации в README.md

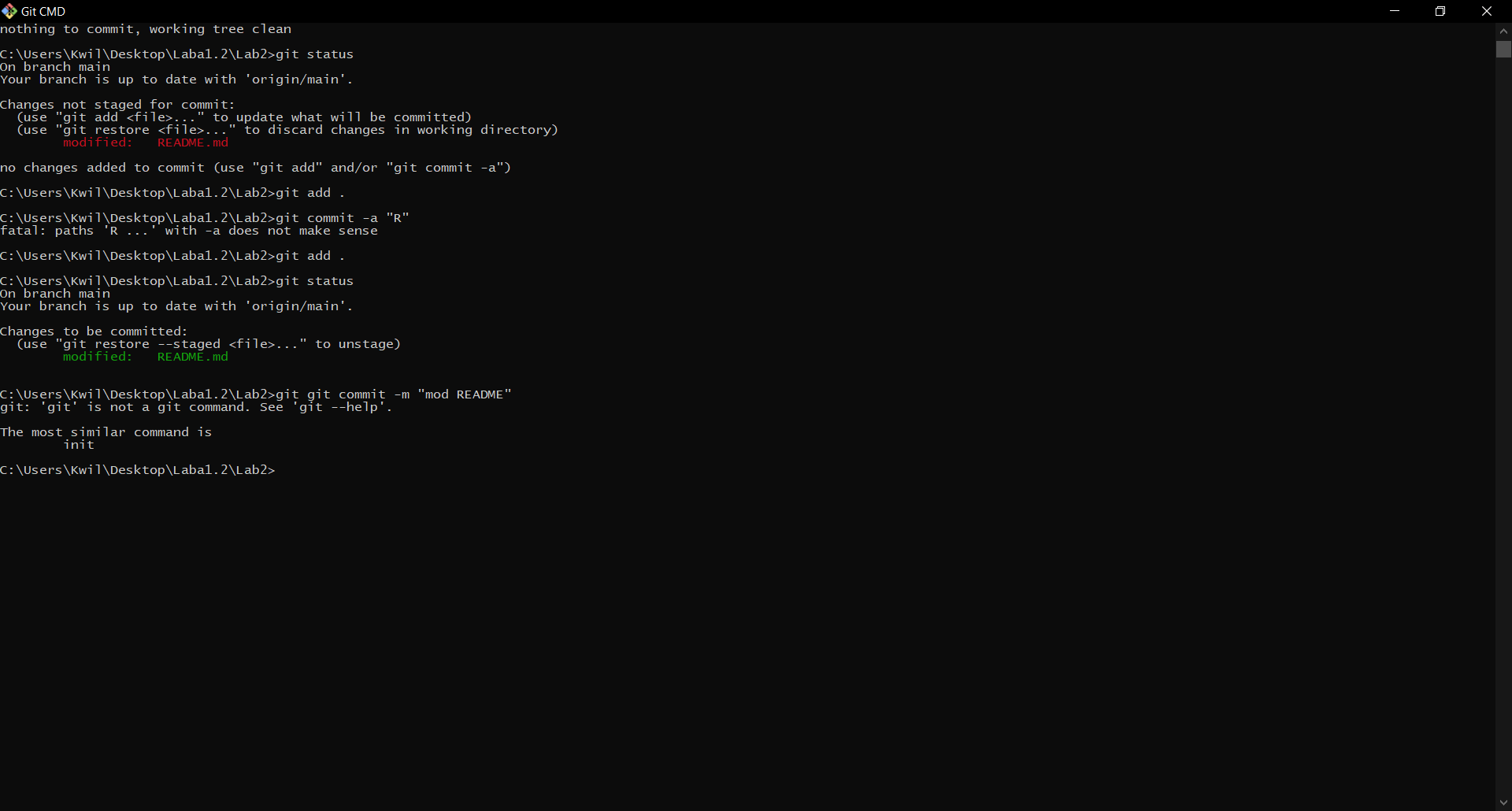


Рисунок 3.2 Коммит файла README.md

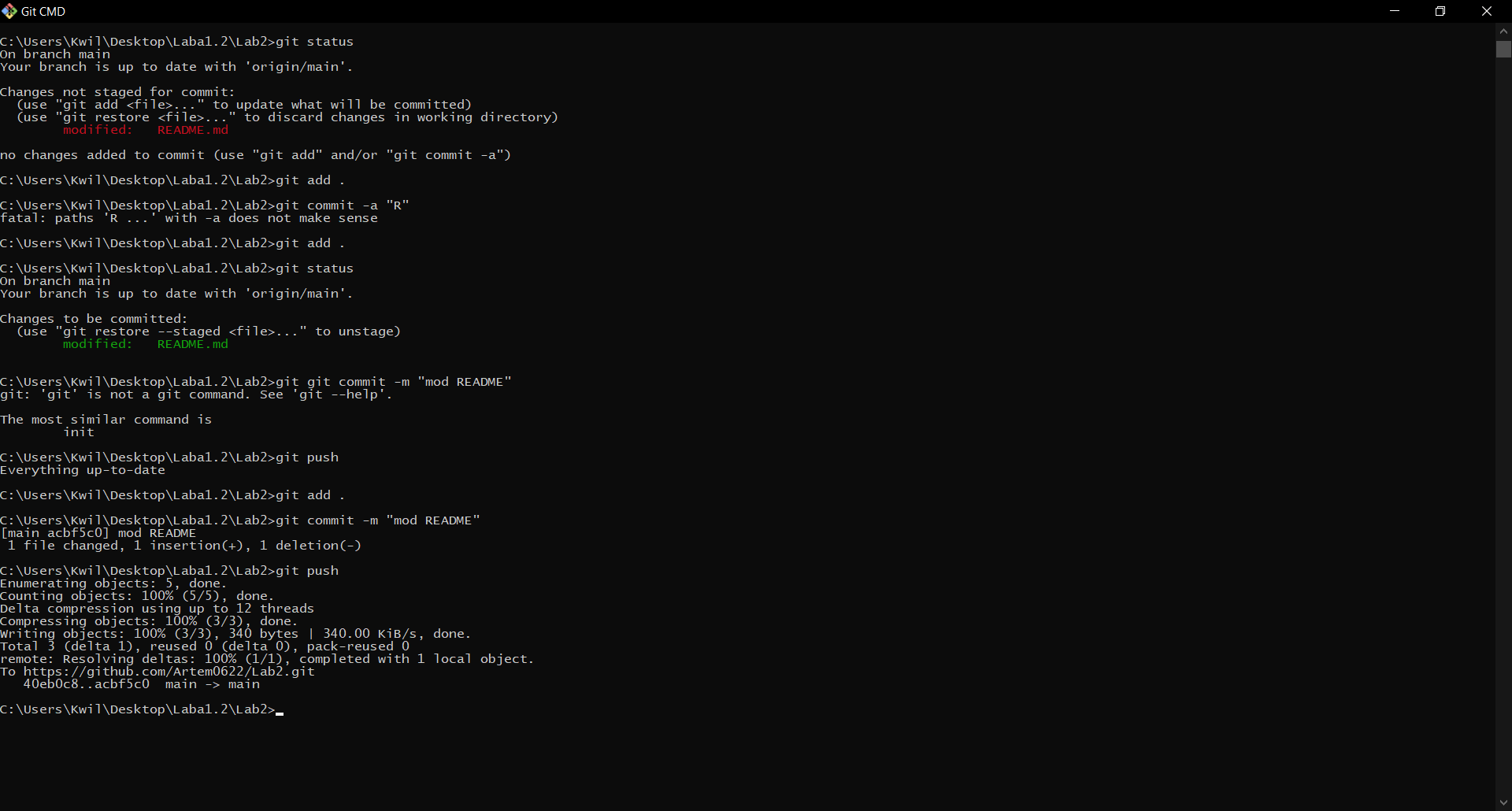


Рисунок 3.3 Пуш коммита

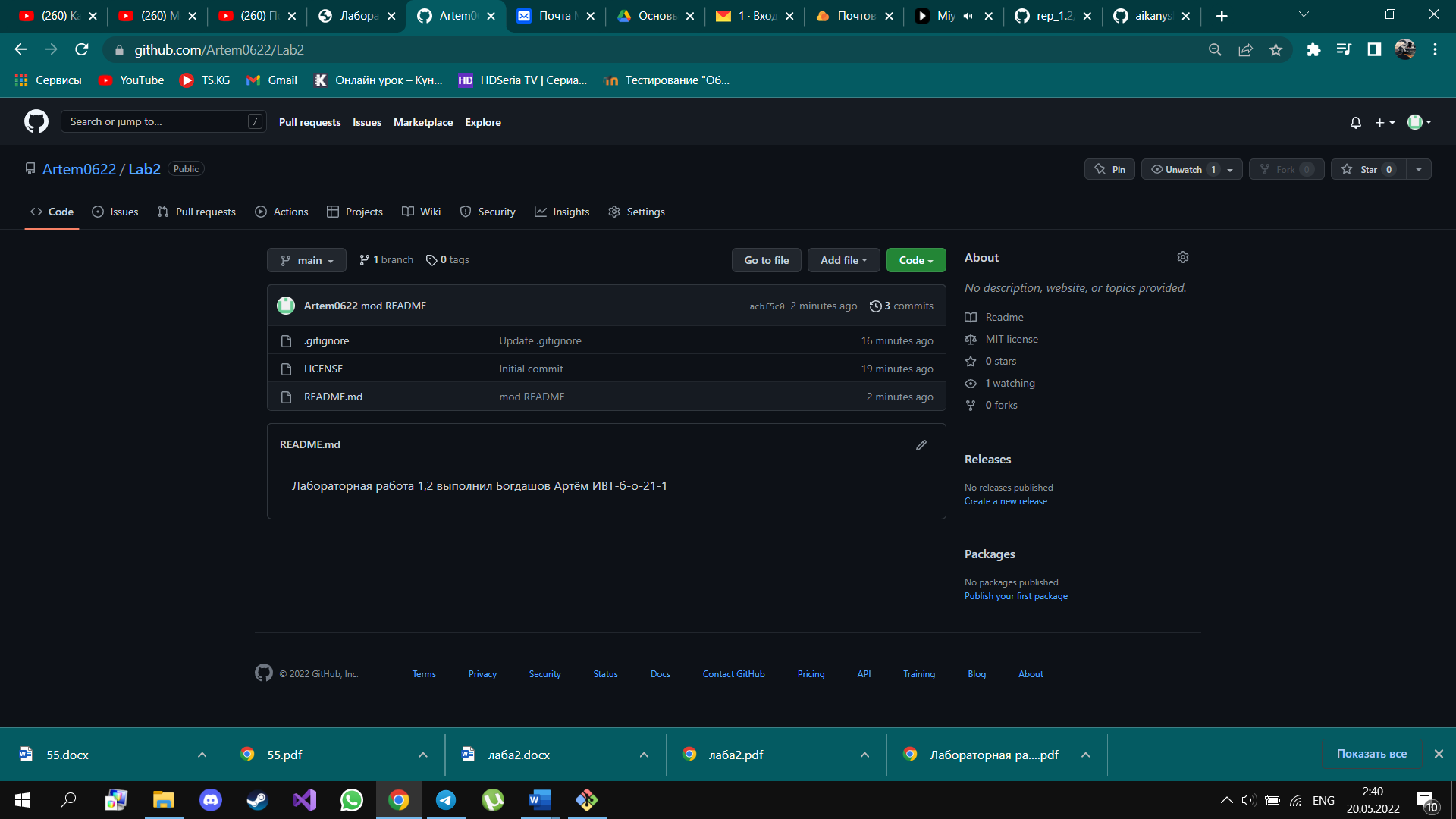


Рисунок 3.4 Изменения на удалённом cервере

4.Написал небольшую программу, сделал коммит и push.

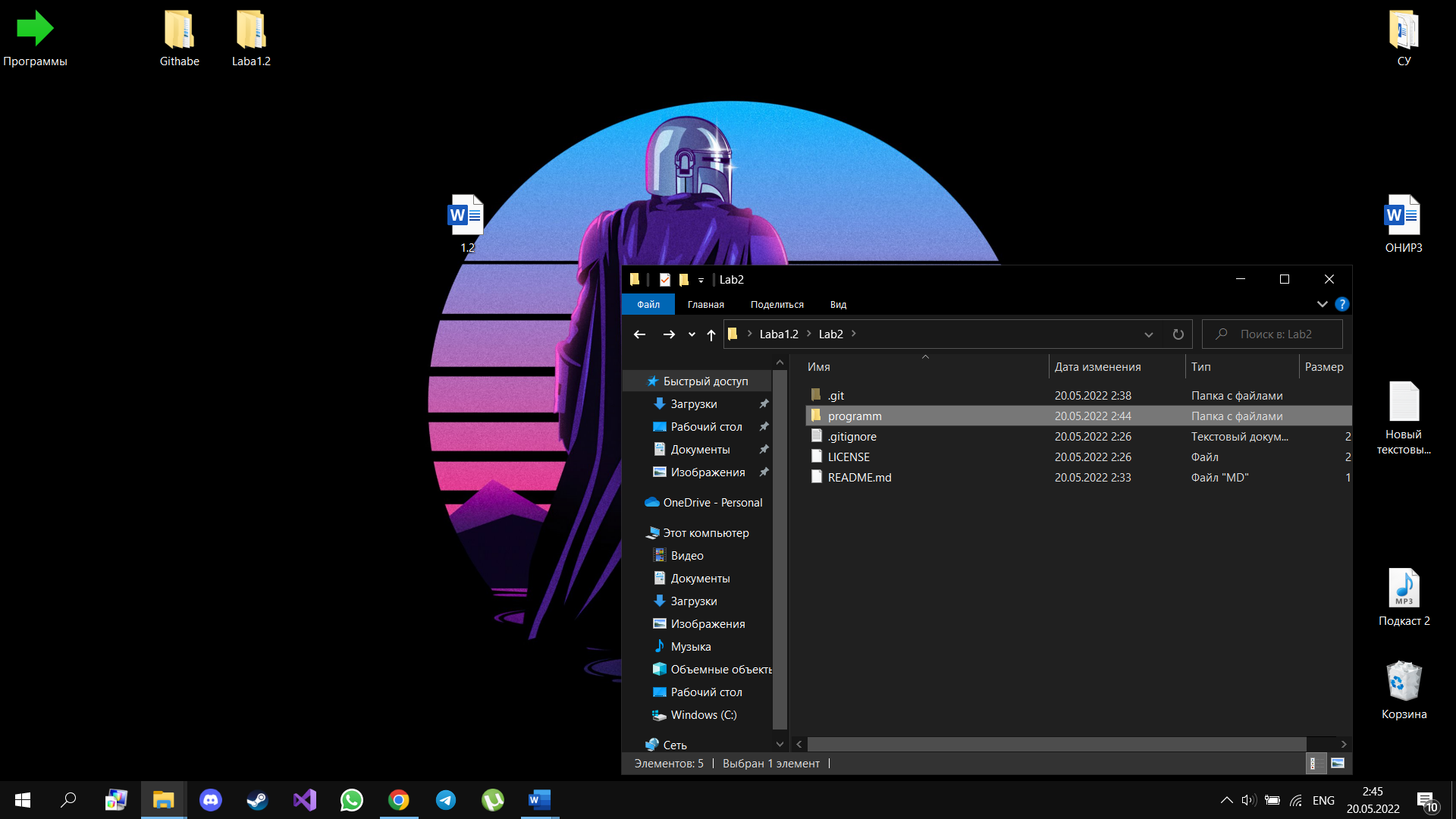


Рисунок 4.1 Добавлена папка с проектом на C++

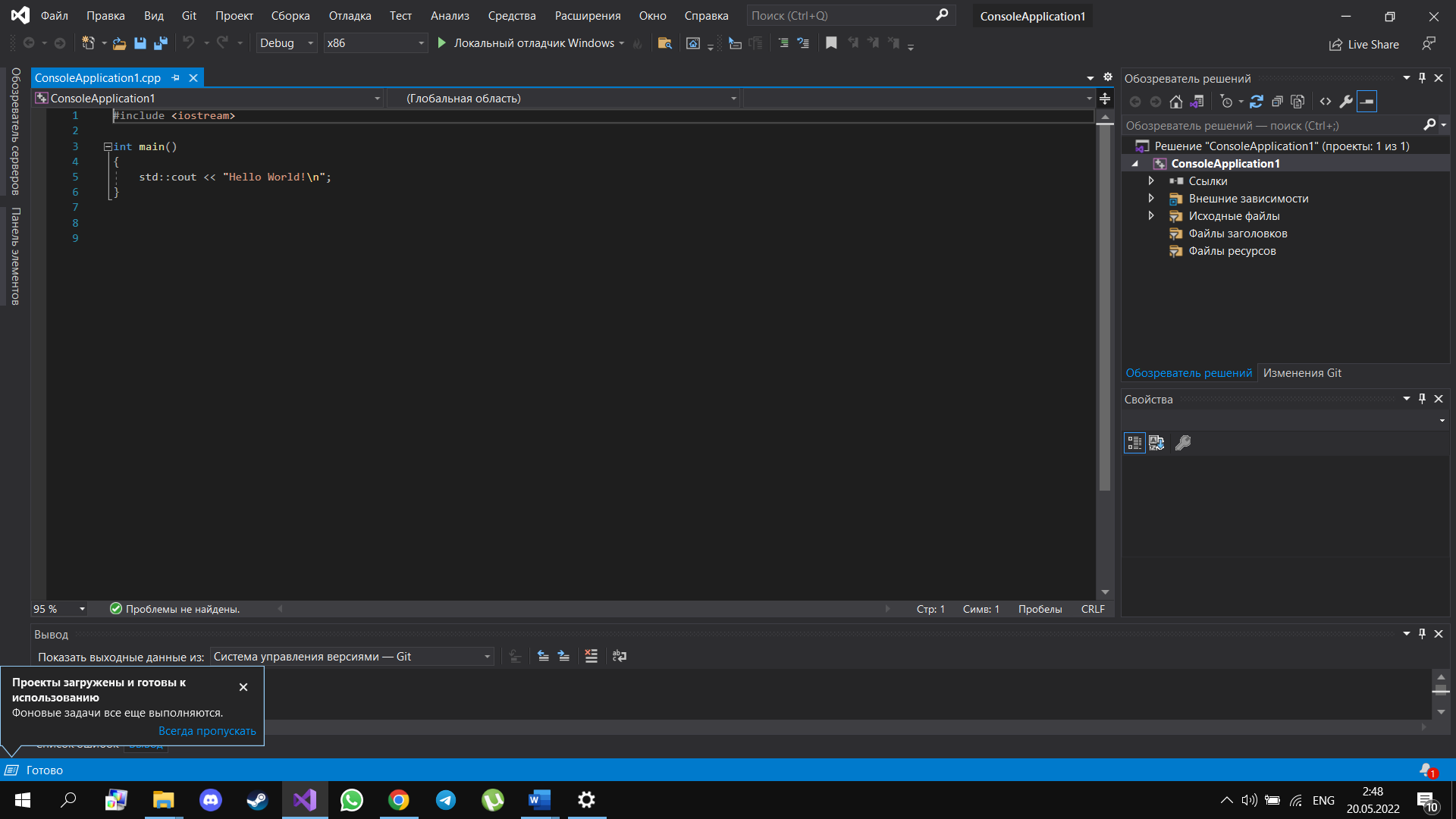


Рисунок 4.2 Изменения в программе

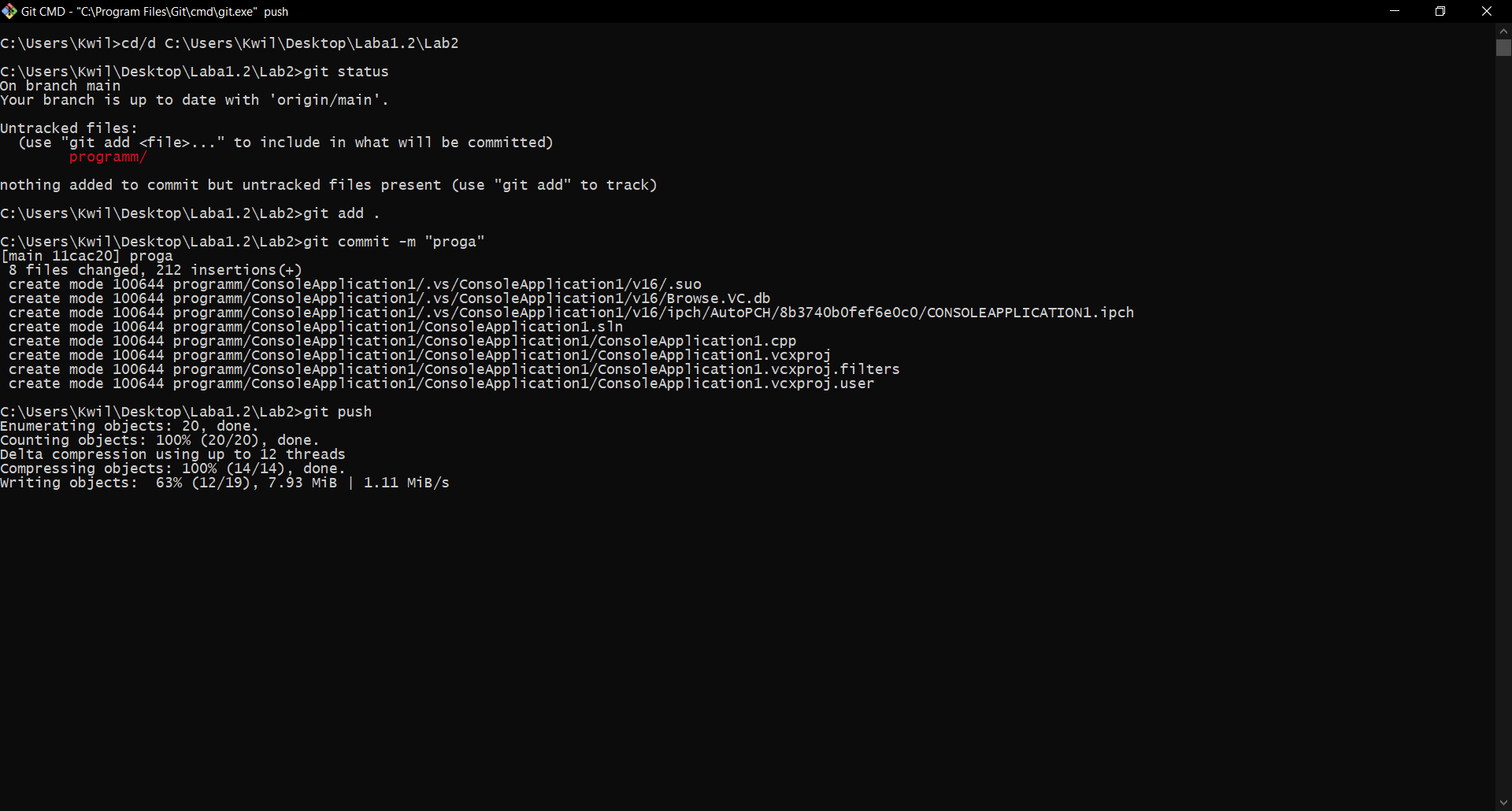


Рисунок 4.3 Коммит и пуш программы на уд. сервер

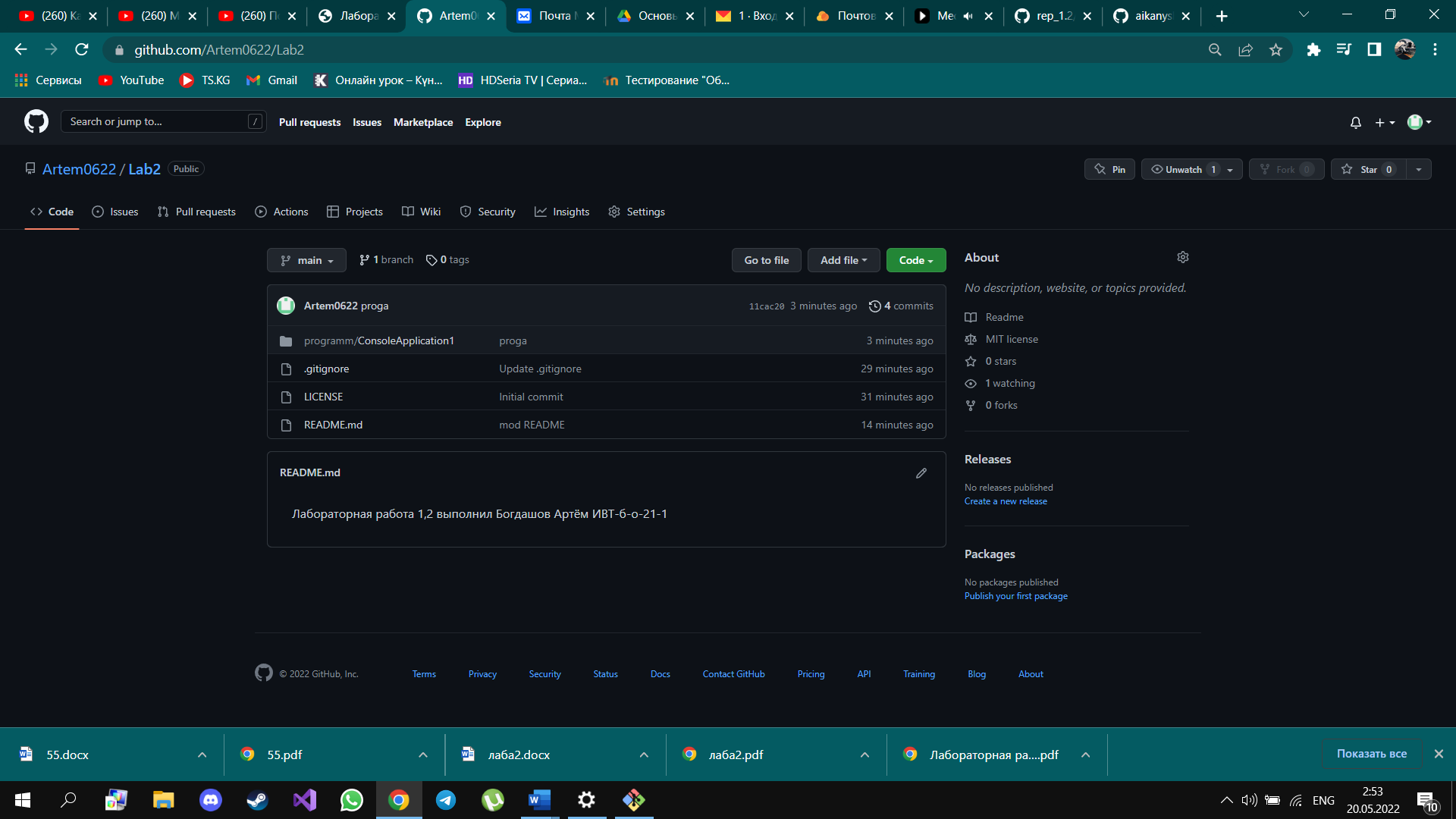


Рисунок 4.4 Изменения на уд. сервере

5.Сделал коммиты в процессе изменения программы, отметил их тегами и запушил на удалённый сервер коммиты затем теги:

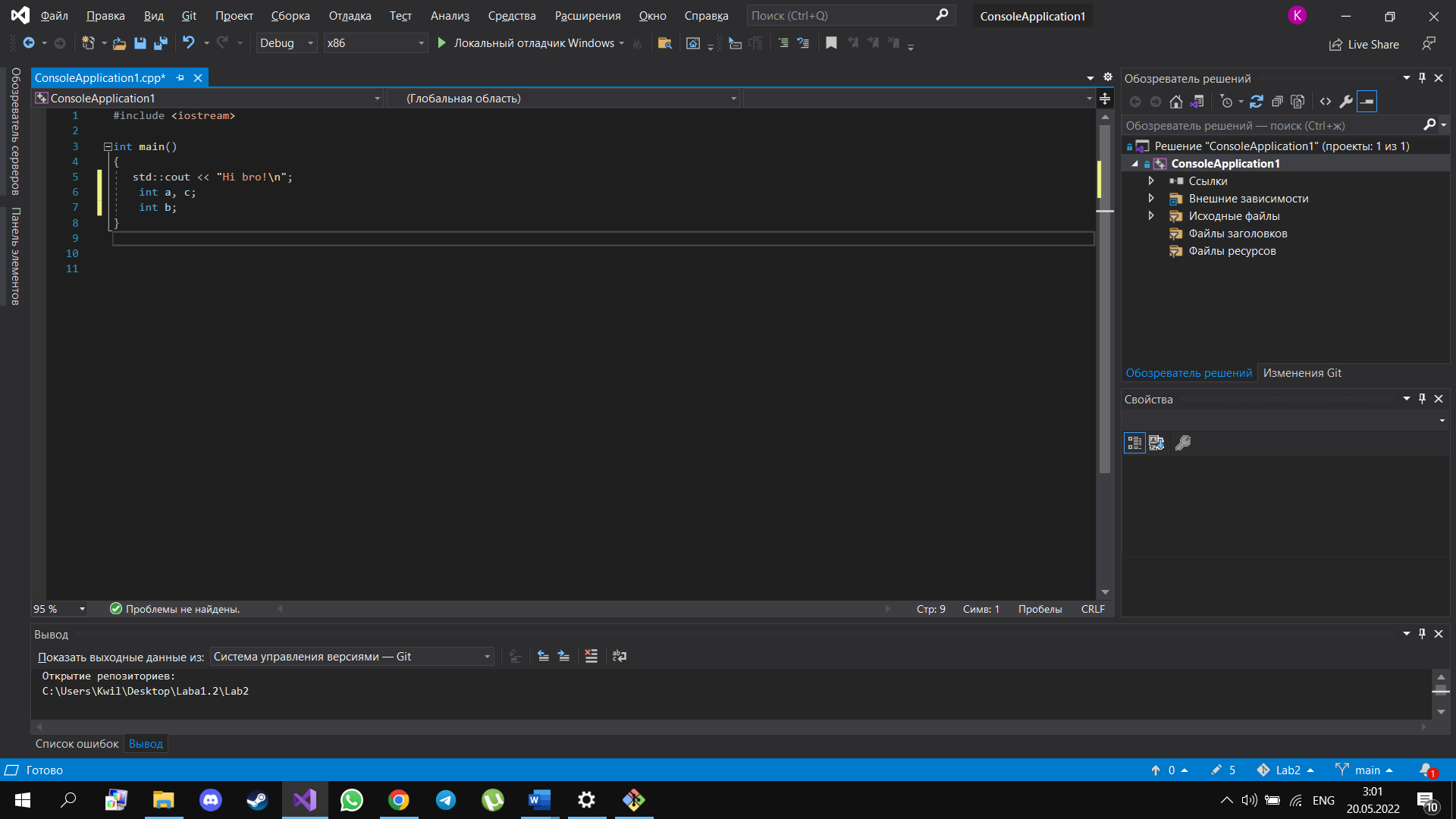


Рисунок 5.1 Изменения в программе

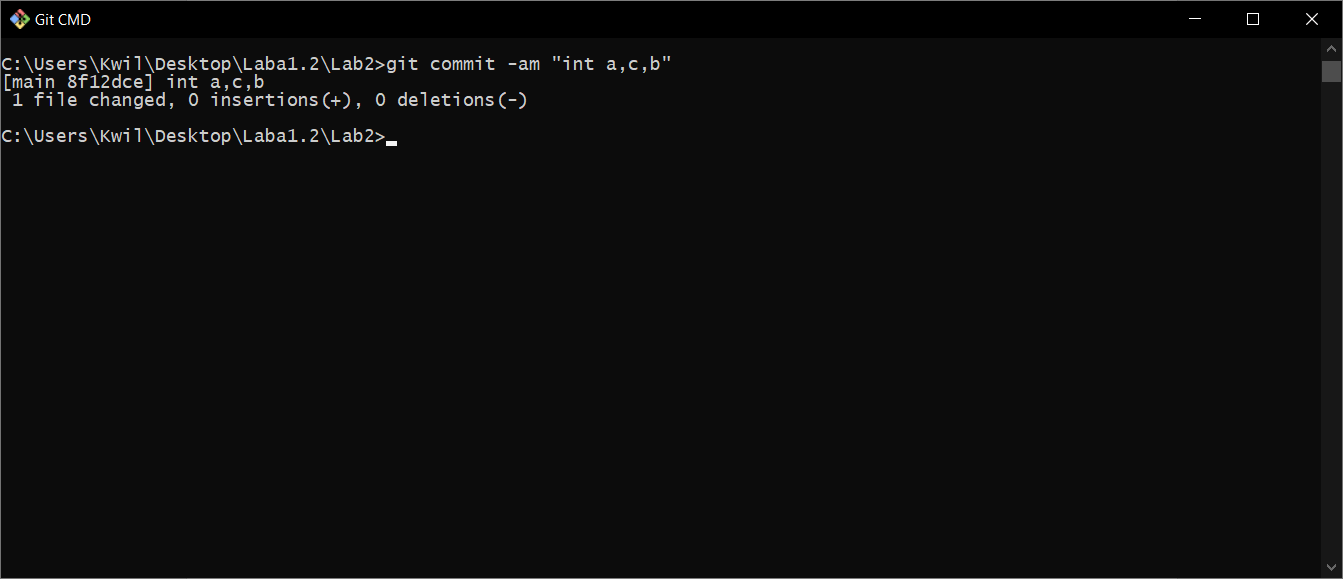


Рисунок 5.2 Коммит изменений

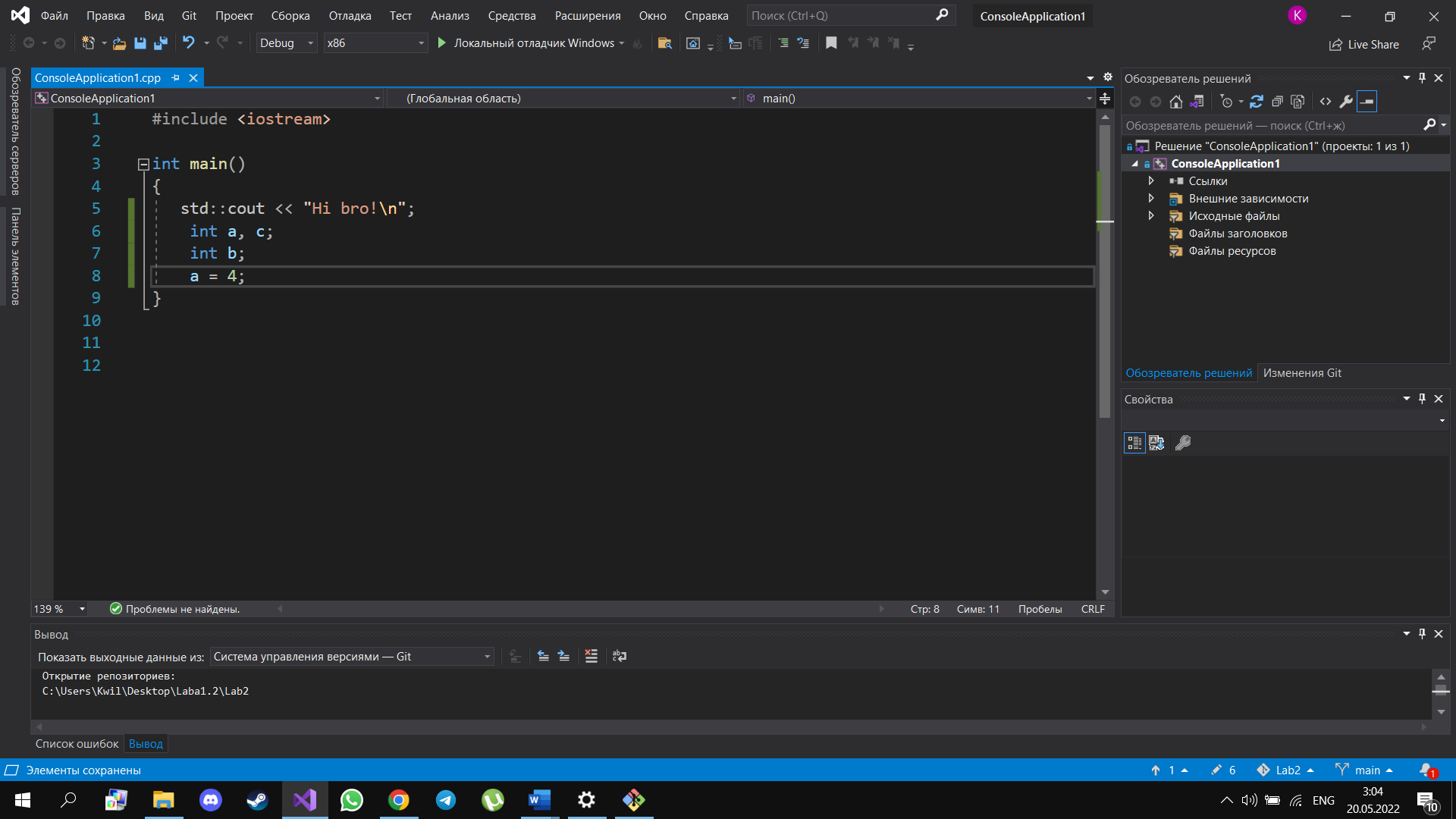


Рисунок 5.3 Изменения в программе



Рисунок 5.4 Коммит изменений

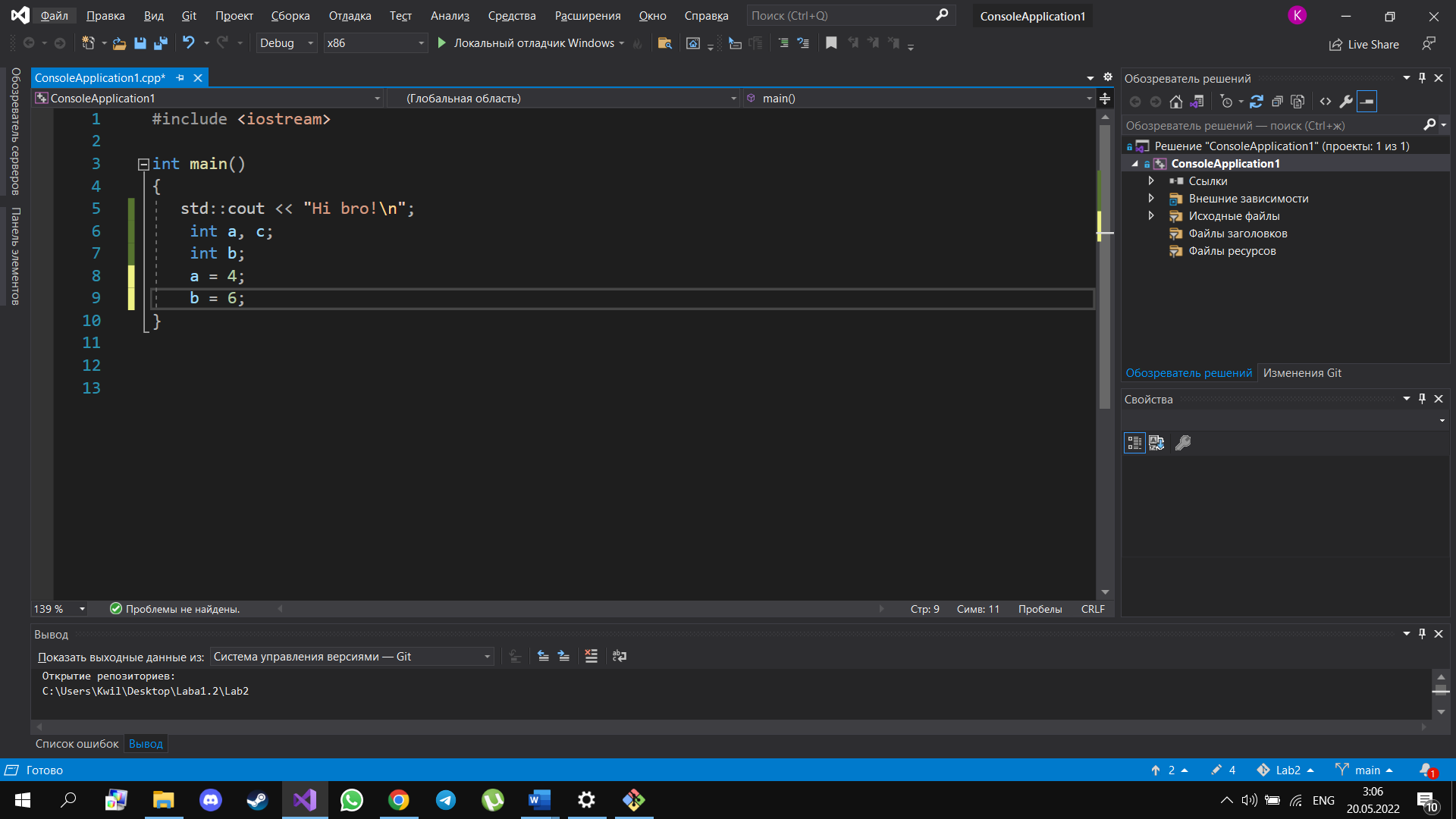


Рисунок 5.5 Изменения в программе

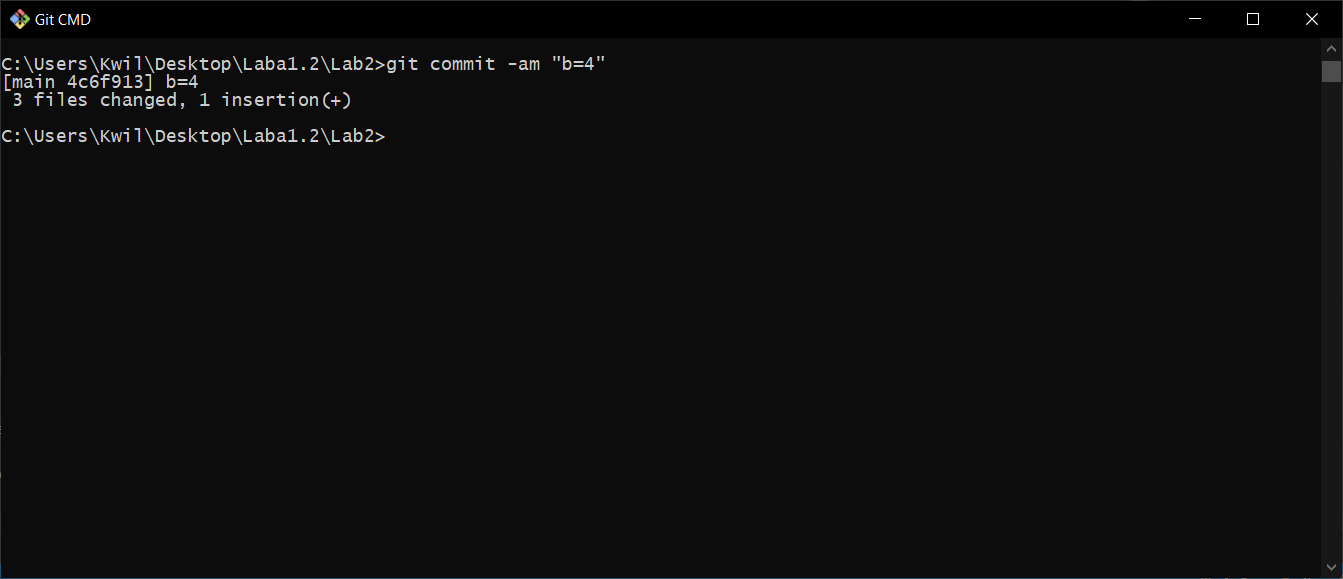


Рисунок 5.6 Коммит изменений

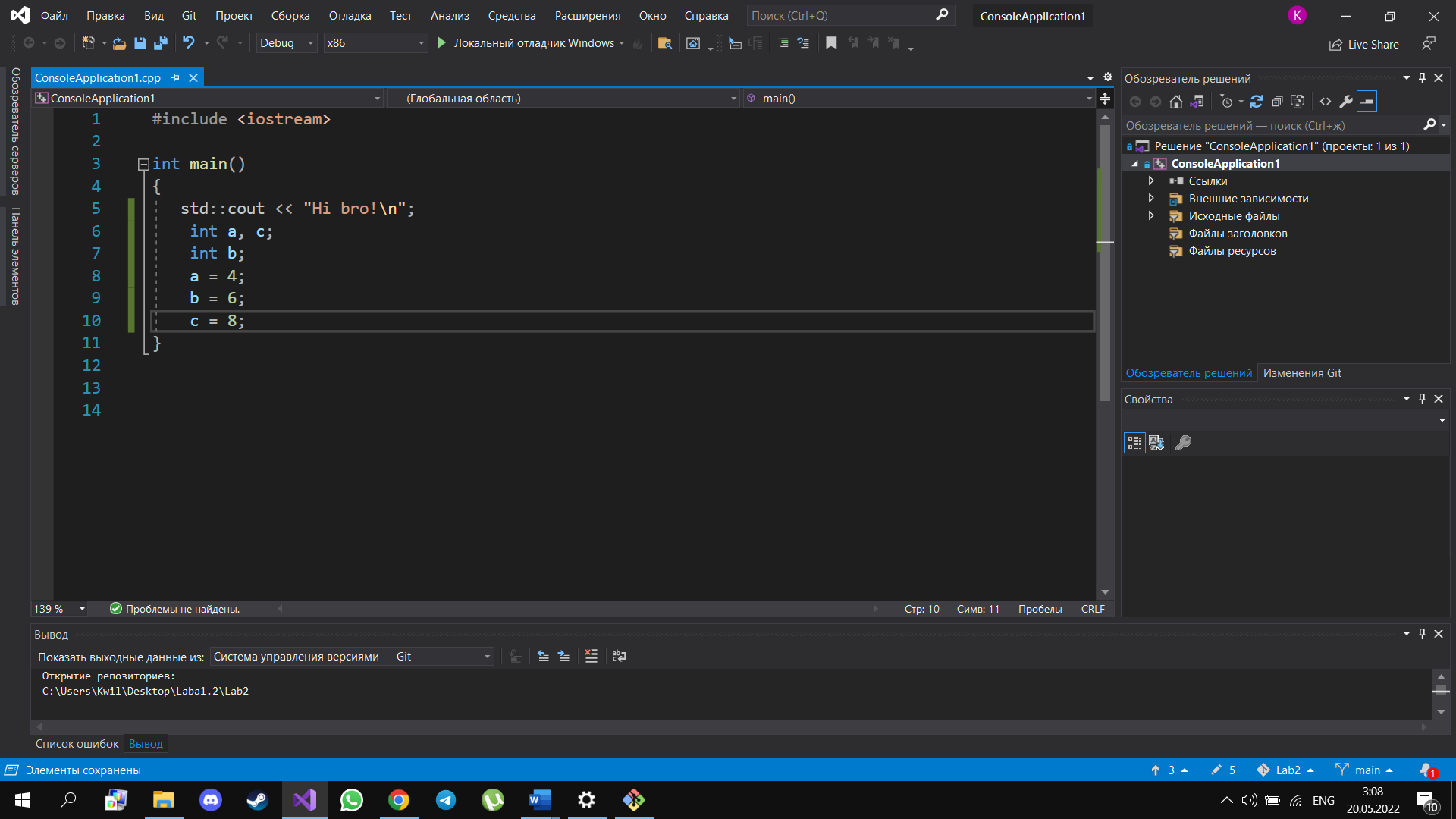


Рисунок 5.7 Изменения в программе

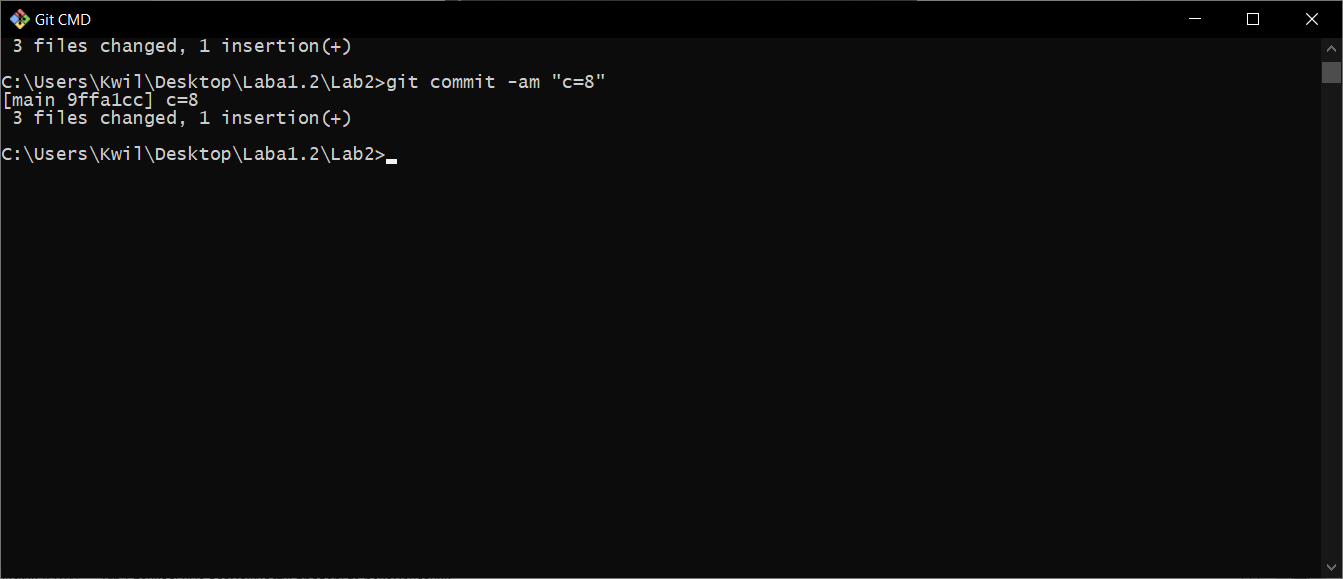


Рисунок 5.8 Коммит изменений

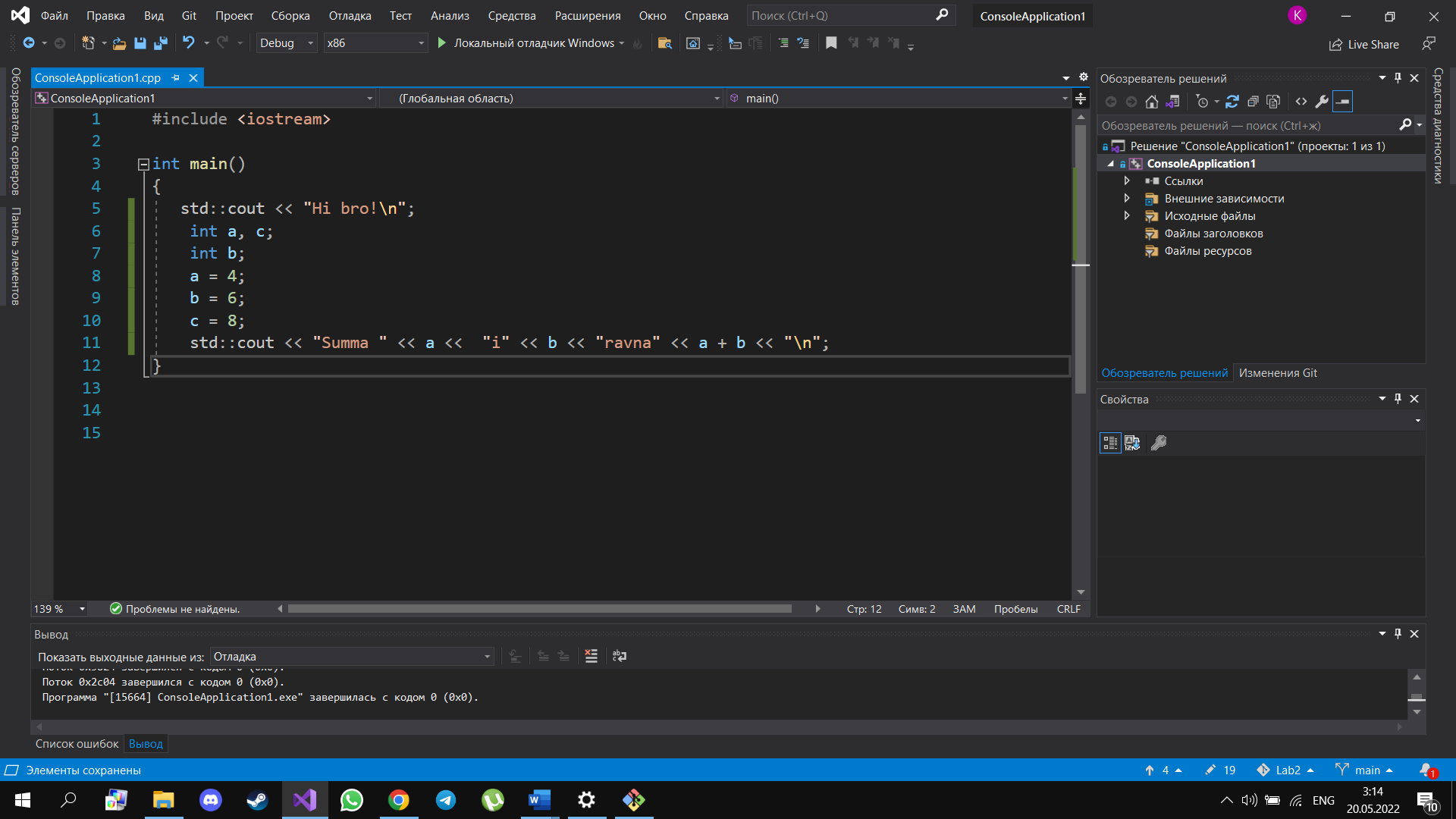


Рисунок 5.9 Изменения в программе



Рисунок 5.10 Коммит изменений

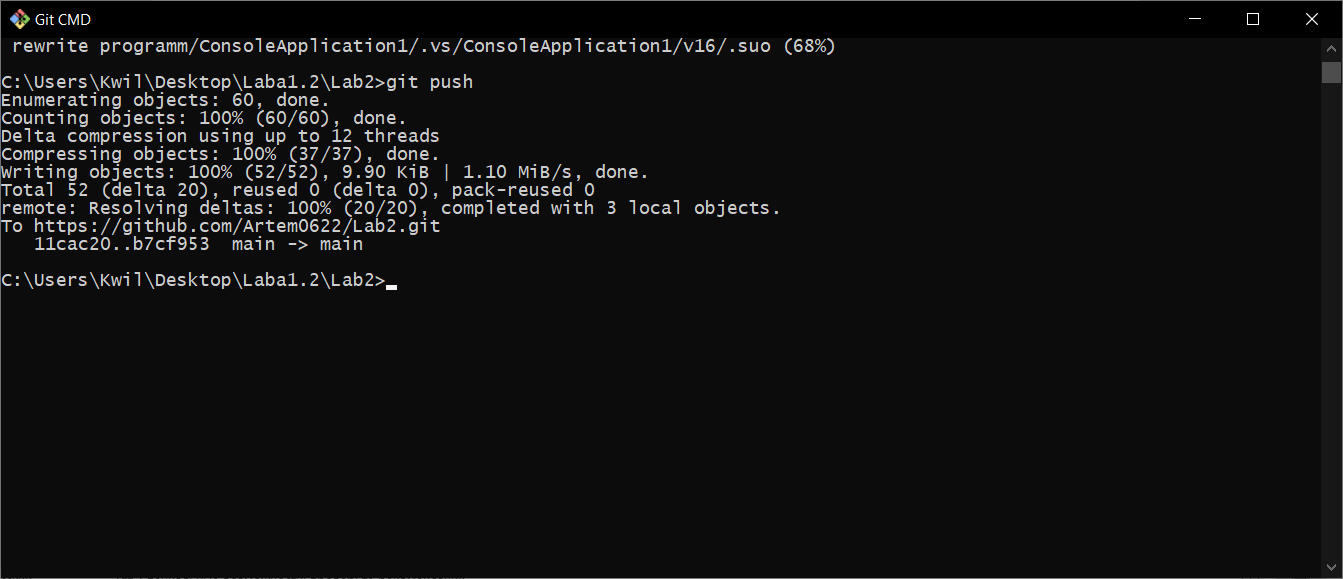


Рисунок 5.11 Пуш изменений

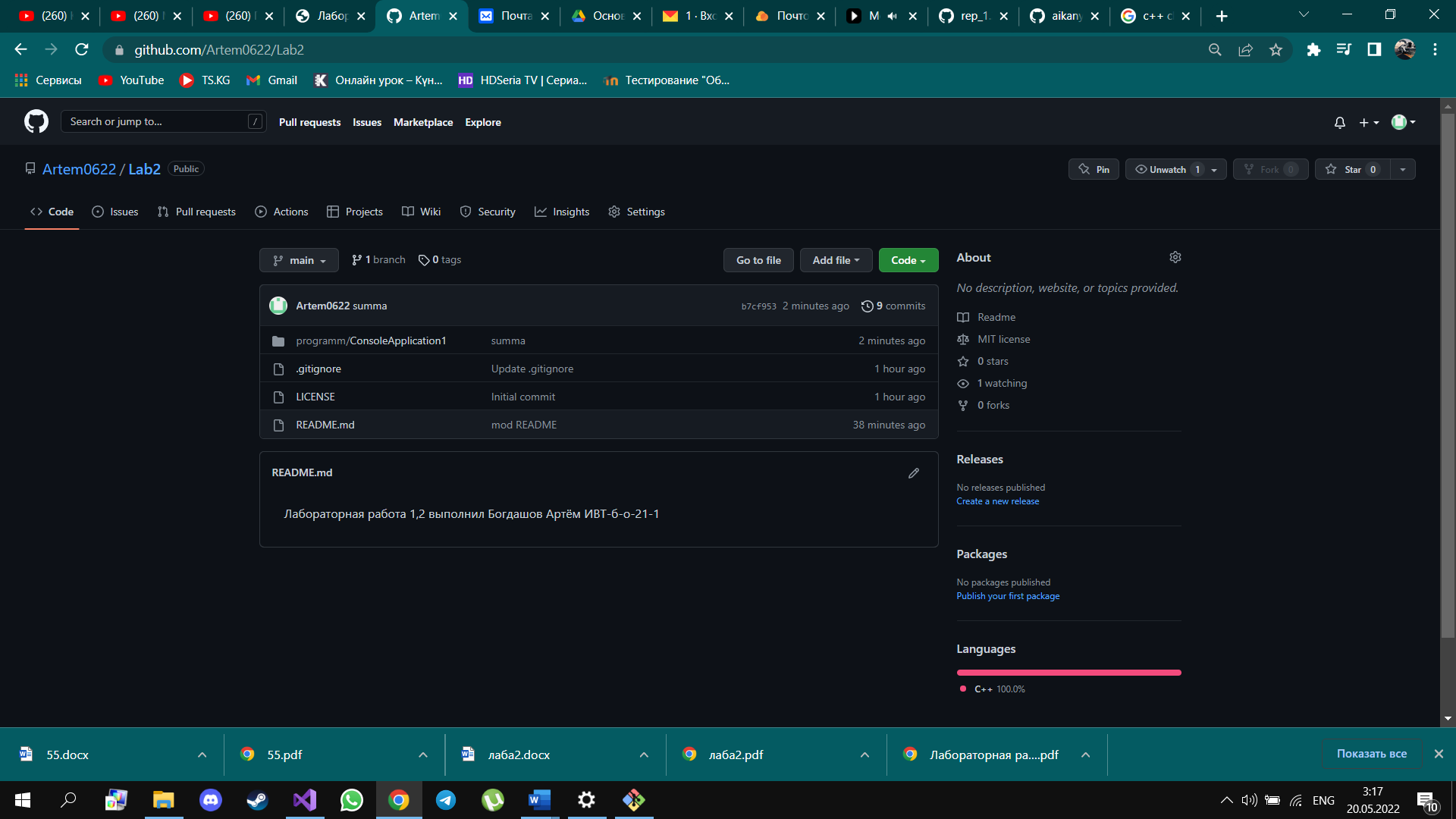


Рисунок 5.12 Изменения на удалённом сервере

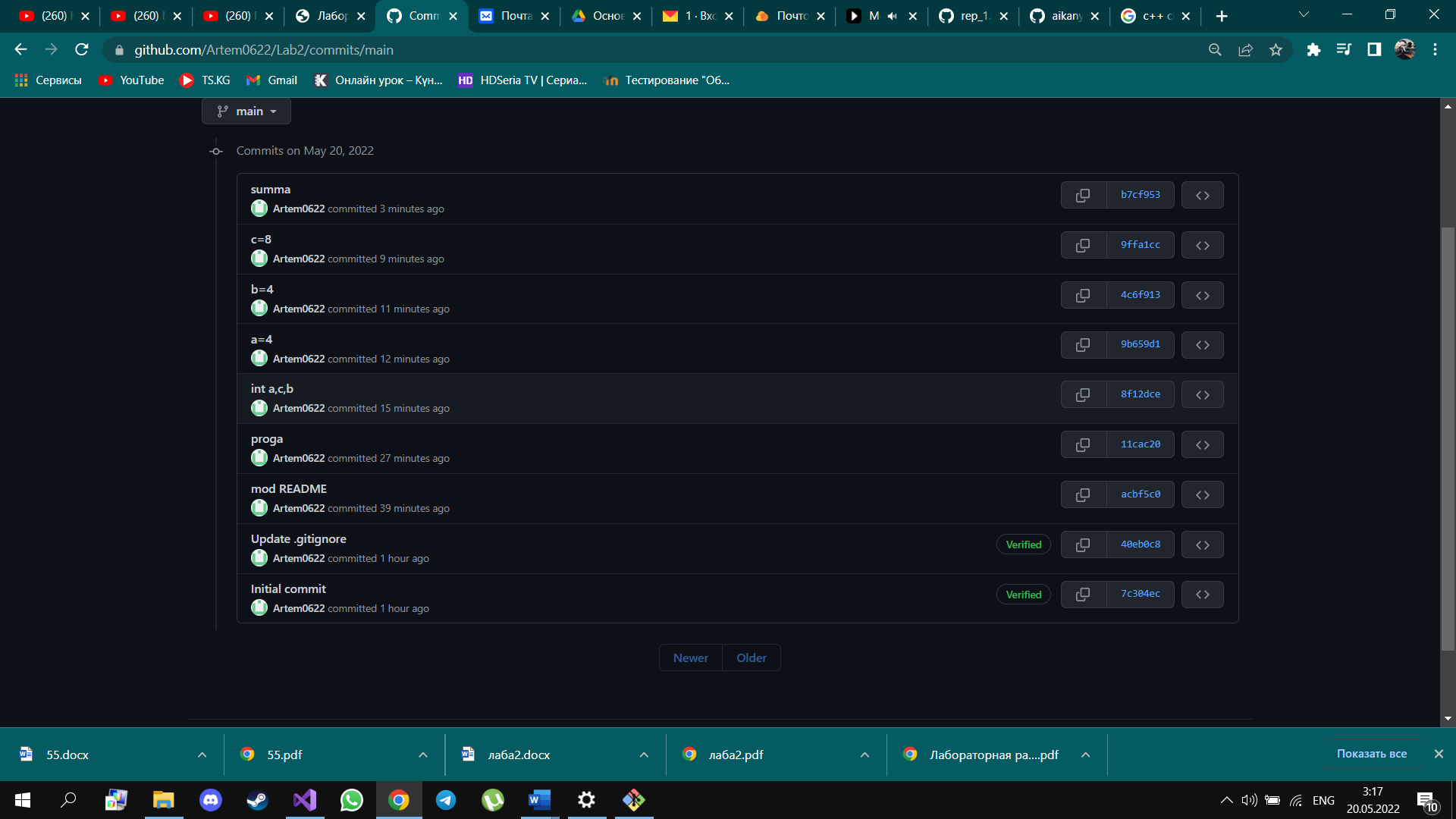


Рисунок 5.13 История коммитов на удалённом сервере

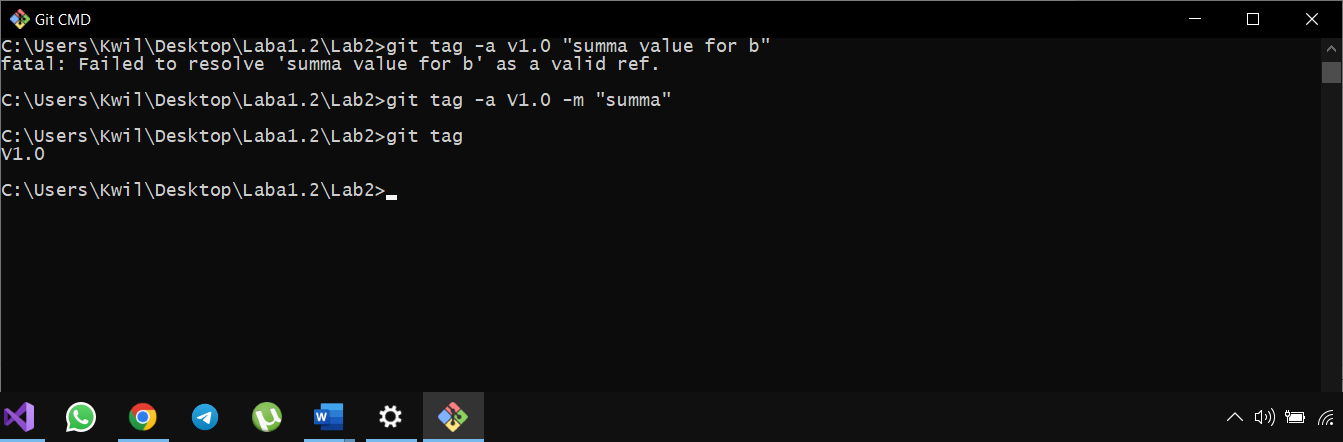
 Рисунок 5.14 Присваивание тега коммиту



Рисунок 5.15 Присваивание тега коммиту

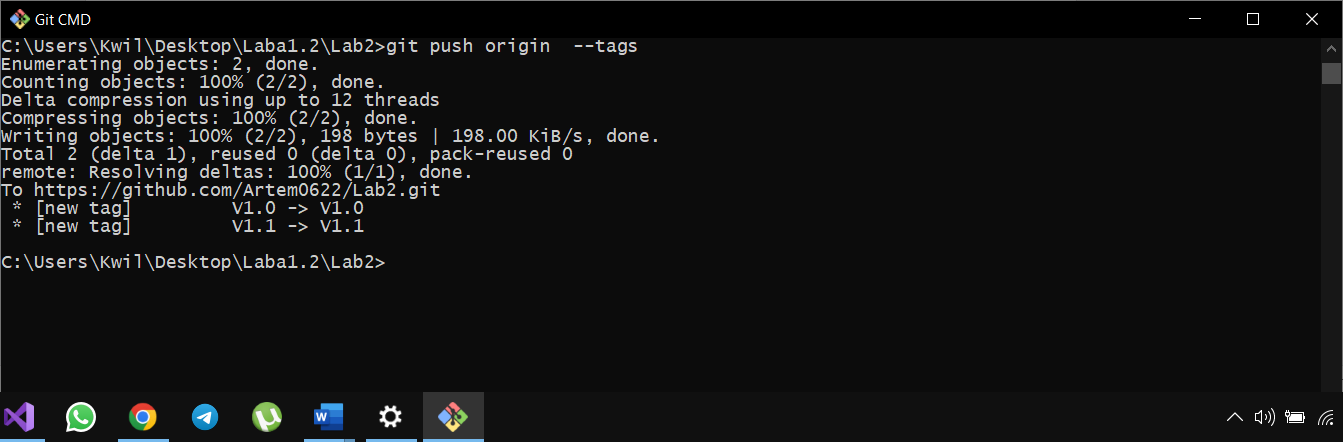


Рисунок 5.16 Push тегов

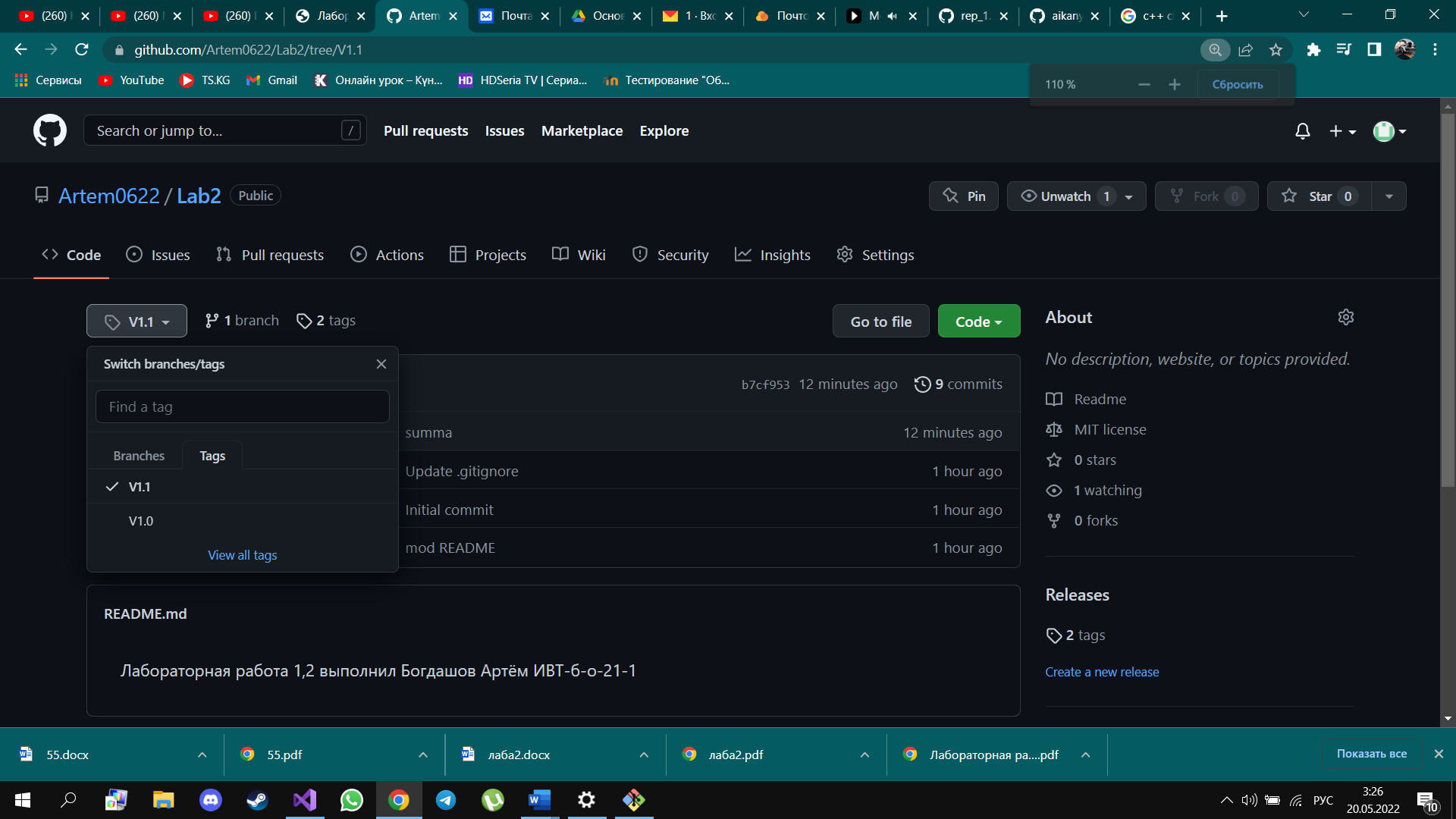


Рисунок 5.17 История тегов на удалённом сервере

6.Просмотрел историю хранилища командой git log:

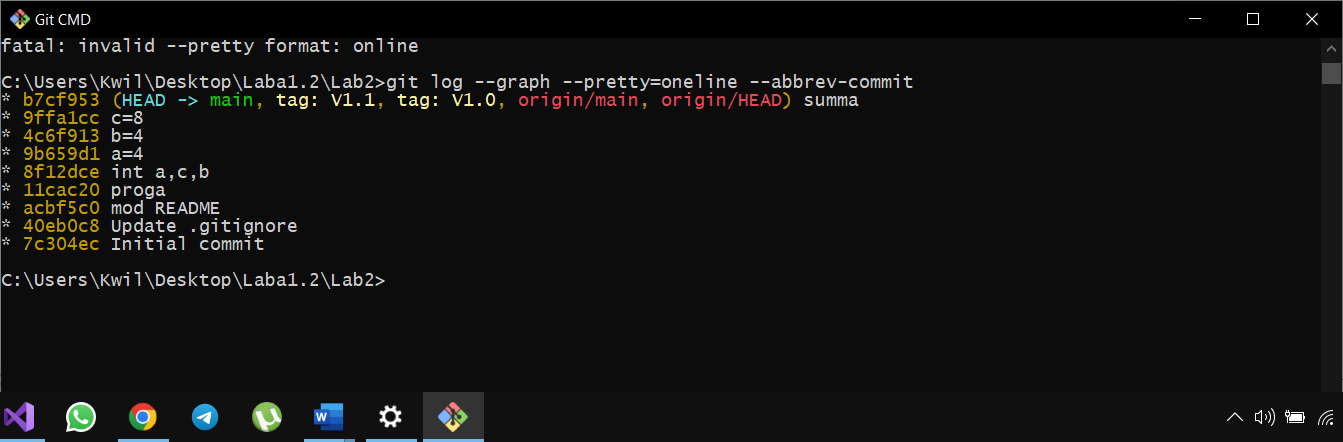


Рисунок 6. История коммитов

7.Просмотрел содержимое коммитов командой git show HEAD, git show HEAD~, git show cb59ad1:

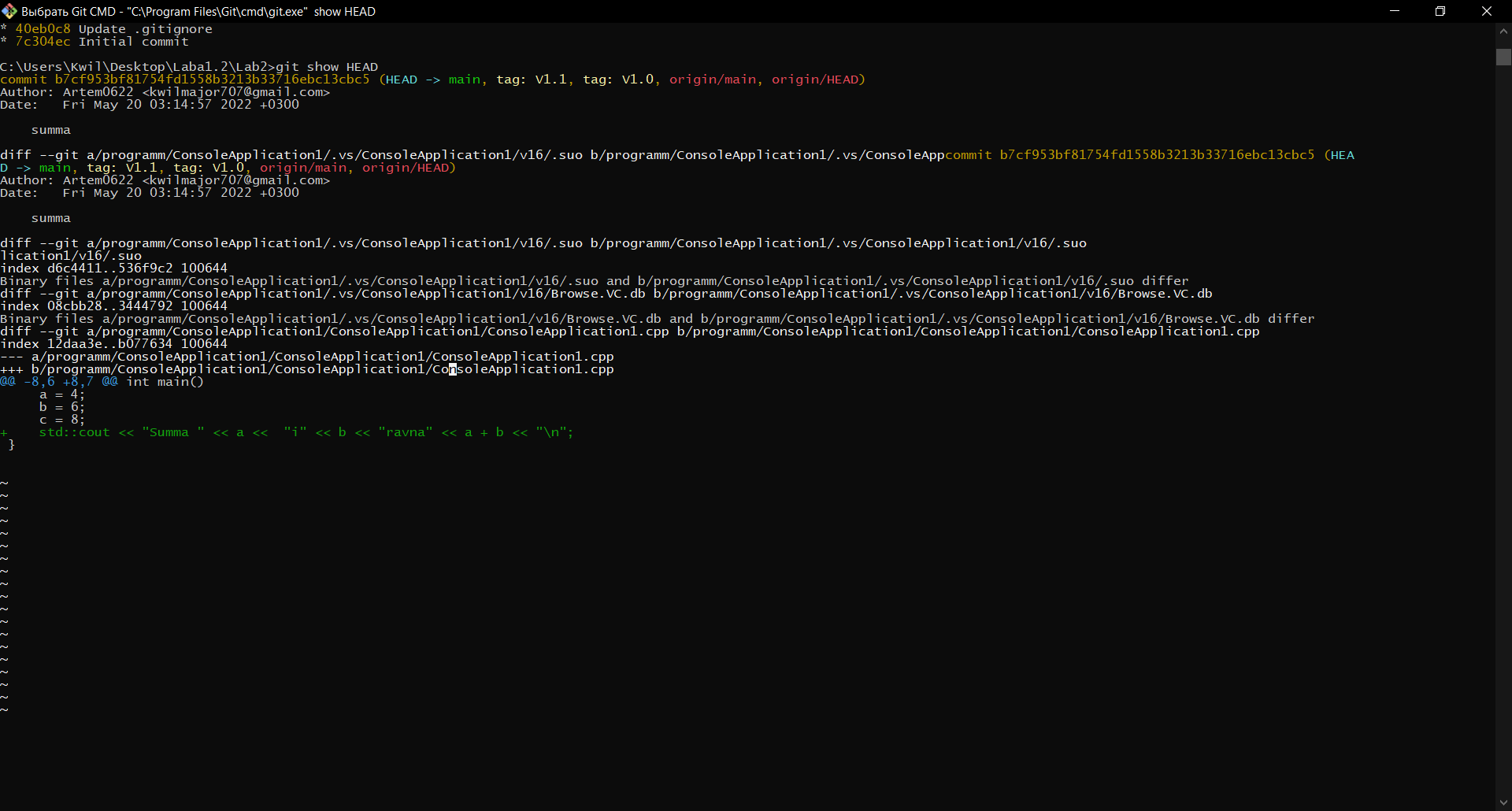


Рисунок 7.1 Содержимое коммитов командами

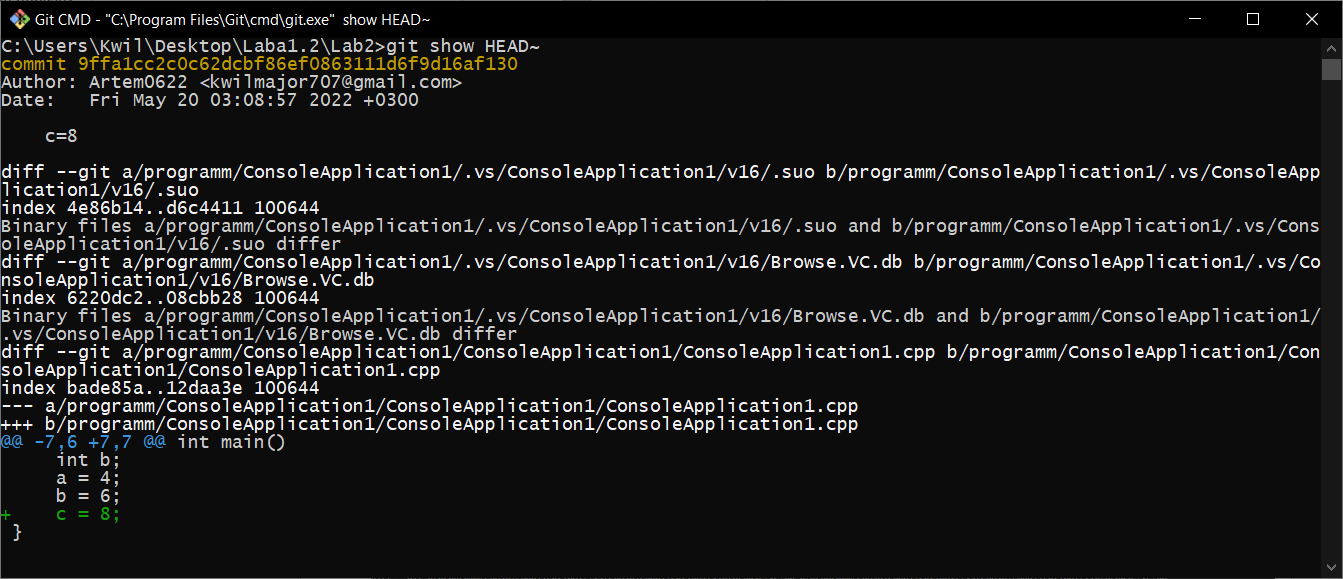


Рисунок 7.2 Содержание коммитов командами

8.Удалил весь код в файле ConsoleApplication1.cpp и сохранил его, затем удалил все несохраненные изменения командой, после этого еще раз удалил весь код в файле и сделал коммит, после чего откатил состояние файла к предыдущей версии.

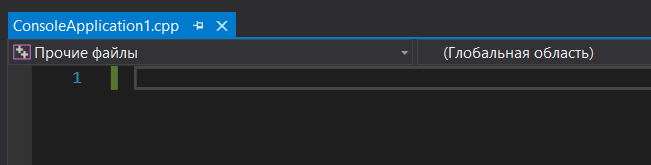


Рисунок 8.1 Удаление кода в файле ConsoleApplication.cpp

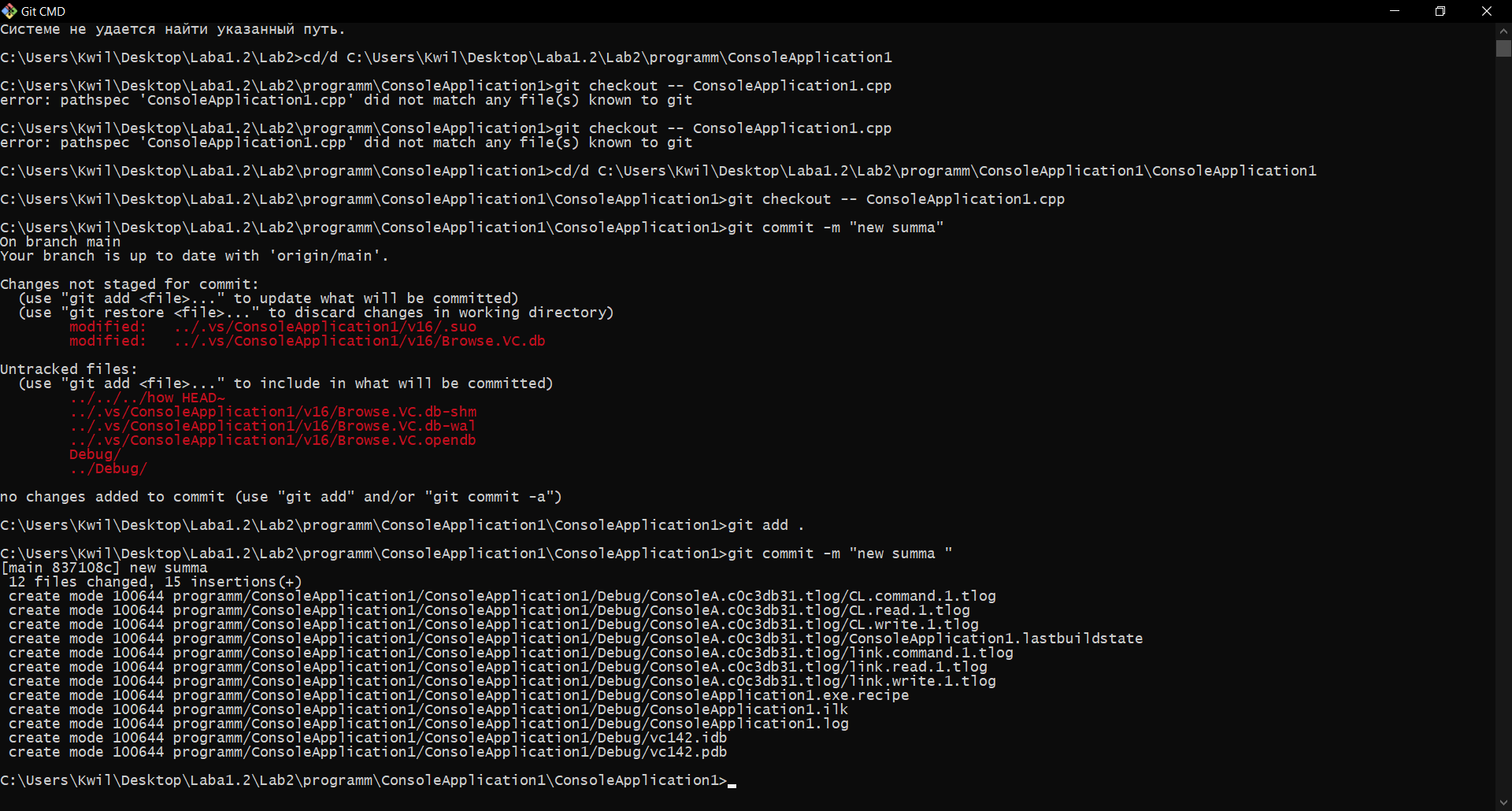


Рисунок 8.2 checkout изменений файла ConsoleApplication.cpp

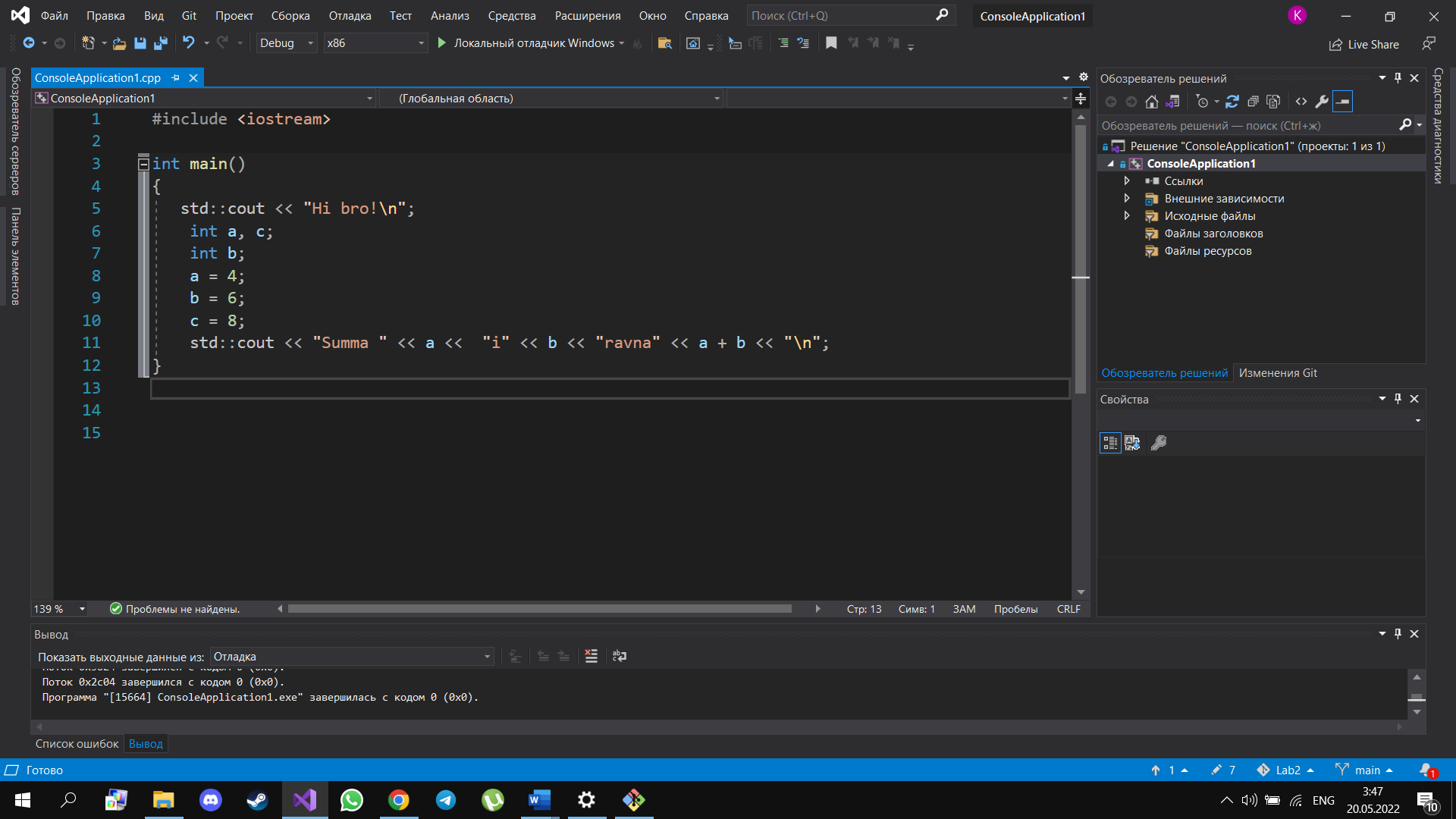


Рисунок 8.3 Изменения в файле с программой после команды

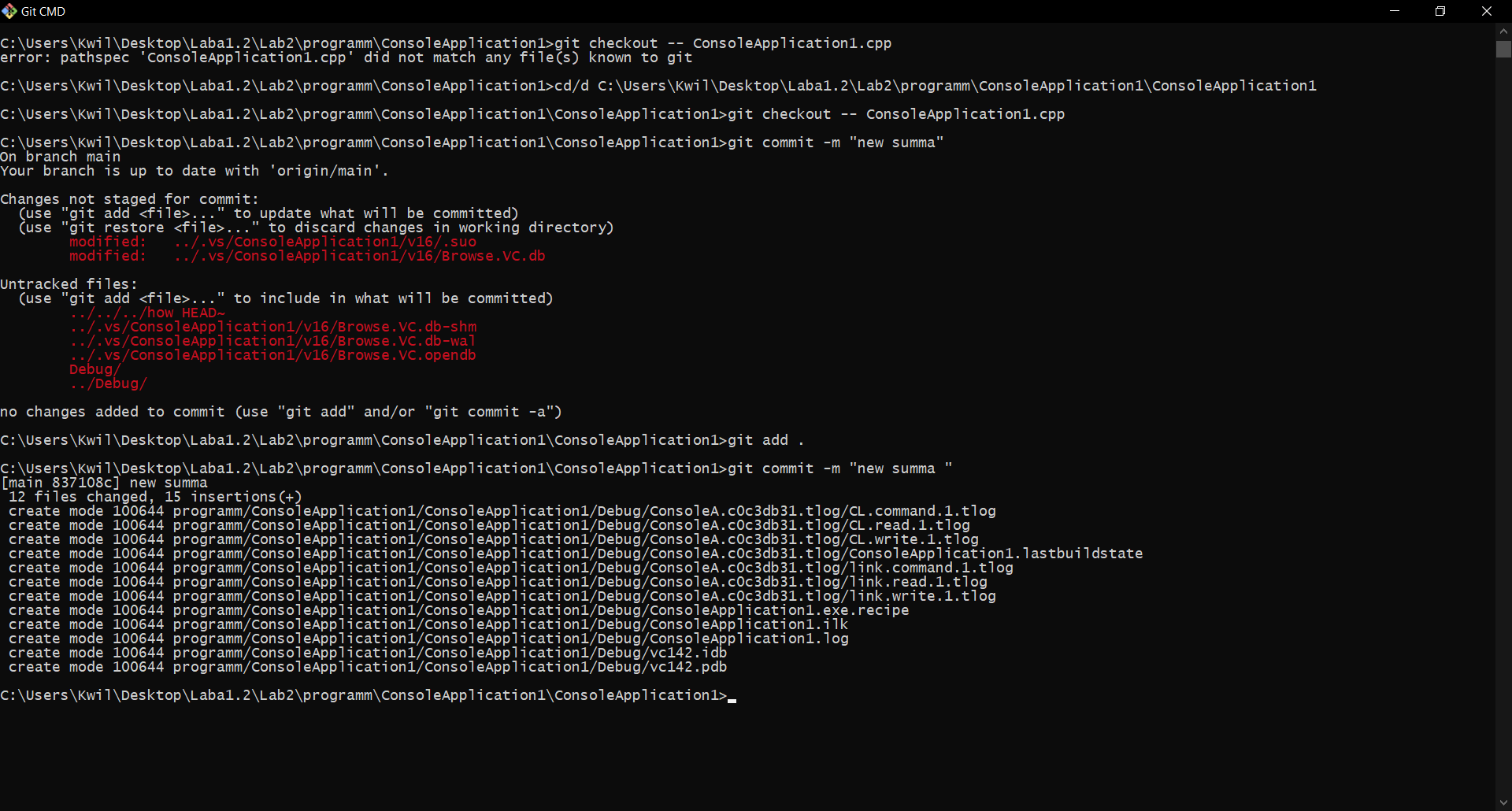


Рисунок 8.4 Коммит изменений

# Вывод: команда git –checkout <FileName> удаляет изменения произошед **Контрольные вопросы и ответы на них:**

Вопросы для защиты работы.

1. Как выполнить историю коммитов в Git? Какие существуют дополни- тельные 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 ком- митов созданных в данном репозитории в обратном хронологическом поряд- ке. 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 ком- мит. Так же вы можете ограничить количество записей ввыводе команды; ис- пользуйте 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 удоб- ным если вы просматриваете большое количество коммитов. К тому же, оп- ции 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 ука- зать 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.

1. 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 мо-жете 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 пара- метрам. 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 по- влекло за собой добавление или удаление указанной строки.

1. 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 из- менений (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 проин- дексировать их и выполнить комманду 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 выпол-нили предыдущий коммит, а еще раз выполнили команду git add и выпол-нили коммит.

1. 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 напомнит вам:

Прямо под текстом «Changes to be committed» говорится: используй-те git reset HEAD <file> для исключения из индекса.

1. Как отменить изменения в файле?

С помощью команды git checkout -- <file>.

1. 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.

1. Как выполнить просмотр удаленных репозиториев данного ло- кального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репо- зиториев, необходимо запустить команду git remote.

Также можно указать ключ -v, чтобы просмотреть адреса для чтения и записи, привязанные к репозиторию. Пример: git remote -v

1. Как добавить удаленный репозиторий для данного локального ре- позитория?

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

1. Как выполнить отправку/получение изменений с удаленного репо- зитория?

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 необходи-мо вручную слить эти данные с вашими, когда вы будете готовы.

Если ветка настроена на отслеживание удалённой ветки, то вы може-те 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 ко- торым вы в данный момент работаете.

Чтобы отправить изменения на удалённый репозиторий необходимо отправить их в удалённый репозиторий. Команда для этого действия про-стая: git push <remote-name> <branch-name>.

1. Как выполнить просмотр удаленного репозитория?

Для просмотра удалённого репозитория, можно использовать ко-манду

git remote show <remote>.

1. Каково назначение тэгов Git?

Теги - это ссылки указывающие на определённые версии ко- да/написанной программы. Они удобно чтобы в случае чего вернутся к нужному моменту. Также при помощи тегов можно помечать важные мо- менты.

1. 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 уда- лённый сервер. Процесс аналогичен отправке веток — достаточно выпол-нить команду git push origin <tagname>. Для отправки всех тегов можно использовать команду git push origin tags.

Для удаления тега в локальном репозитории достаточно выполнить команду git tag –d <tagname>. Например, удалить созданный ранее легко- весный 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.

1. 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.

шие с файлом в репозитории до коммита.