

**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-21-1

Коновалова В.Н. « 15» сентября 2022г.

Подпись студента\_\_\_\_\_

Работа защищена « »\_\_\_\_\_20\_\_г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2022

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

<https://github.com/VictoriaKonovalova/Lr1>

## Ответы на контрольные вопросы

### 1. Что такое СКВ и каково ее назначение?

Программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

### 2. В чем недостатки локальных и централизованных СКВ?

Главный минус централизованных СКВ – уязвимость централизованного сервера. Временное выключение сервера (пусть даже на час) останавливает работу программистов, они не могут ни сохранять новые варианты версий, ни взаимодействовать между собой. В случае же повреждения диска, на котором хранится центральная база данных, все наработки по проекту теряются безвозвратно.

С локальными системами контроля версий та же история: если все данные по проекту «лежат» в одном месте, вы можете лишиться их сразу в один момент. Также ее проблемой является основное свойство — локальность. Она совершенно не предназначена для коллективного использования.

### 3. К какой СКВ относится Git?

Распределенная СКВ. Суть их работы состоит в выгрузке клиентам не только версий файлов с последними изменениями, а всего репозитория. В большинстве систем доступно для хранения данных сразу нескольких удаленных репозиториях.

#### 4. В чем концептуальное отличие Git от других СКВ?

Главное отличие Git'a от любых других СКВ (например, Subversion и ей подобных) — это то, как Git смотрит на свои данные. В принципе, большинство других систем хранит информацию как список изменений (патчей) для файлов. Эти системы (CVS, Subversion, Perforce, Bazaar и другие) относятся к хранимым данным как к набору файлов и изменений, сделанных для каждого из этих файлов во времени. Вместо этого Git считает хранимые данные набором слепков небольшой файловой системы.

#### 5. Как обеспечивается целостность хранимых данных в Git?

Перед сохранением любого файла Git вычисляет контрольную сумму, и она становится индексом этого файла. Поэтому невозможно изменить содержимое файла или каталога так, чтобы Git не узнал об этом. Эта функциональность встроена в сам фундамент Git и является важной составляющей его философии. Механизм, используемый Git для вычисления контрольных сумм, называется SHA-1 хеш. Это строка из 40 шестнадцатеричных знаков (0-9 и a-f), которая вычисляется на основе содержимого файла или структуры каталога, хранимого Git. Git сохраняет всё не по именам файлов, а по хешам их содержимого.

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

Git имеет три основных состояния, в которых могут находиться ваши файлы: изменённые, индексированные и зафиксированные.

Изменённый означает, что вы изменили файл, но ещё не зафиксировали его в своем локальном репозитории.

Индексированный - это изменённый файл, текущую версию которого вы отметили для включения в следующий коммит (для фиксации в своём локальном репозитории).

Зафиксированный означает, что файл уже сохранён в вашем локальном репозитории.

#### 7. Что такое профиль пользователя в GitHub?

Профиль пользователя – это аккаунт, в котором человек может хранить различные проекты, версии этих проектов, предоставлять другим пользователям возможность предлагать изменения, а также самому предлагать улучшения чужих проектов, предоставлять удаленный доступ к файлам.

#### 8. Какие бывают репозитории в GitHub?

Локальный репозиторий – это репозиторий, который хранится на нашей машине, в рабочей папке проекта. Это та самая скрытая папка `.git`

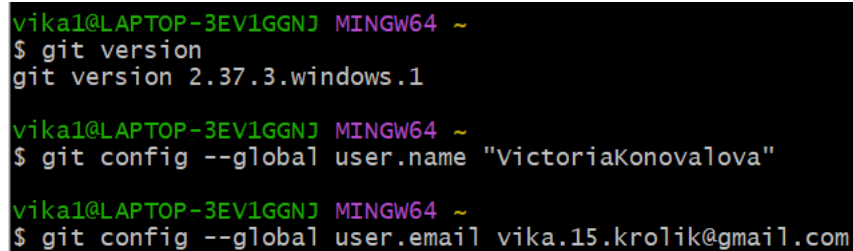
Удаленный репозиторий - это репозиторий, который хранится в облаке, на сторонних сервисах, специально созданных под работу с проектами `git`.

#### 9. Укажите основные этапы модели работы с GitHub.

В начале создается репозиторий на GitHub. Далее создается запрос на извлечение файлов и в следствии локальная копия всего репозитория со всеми версиями файлов, которая размещается на рабочем устройстве. Потом пользователь производит различные изменения в проекте и фиксирует их в удаленный репозиторий.

#### 10. Как осуществляется первоначальная настройка Git после установки?

Нужно ввести свое имя пользователя и адрес электронной почты, можно также проверить версию установленного продукта – это и будет первоначальной настройкой. Сделать это можно с помощью командной строки следующим образом:



```
vika1@LAPTOP-3EV1GGNJ MINGW64 ~  
$ git version  
git version 2.37.3.windows.1  
  
vika1@LAPTOP-3EV1GGNJ MINGW64 ~  
$ git config --global user.name "VictoriaKonovalova"  
  
vika1@LAPTOP-3EV1GGNJ MINGW64 ~  
$ git config --global user.email vika.15.krolik@gmail.com
```

Рисунок 1.1– первоначальная настройка Git

#### 11. Опишите этапы создания репозитория в GitHub.

Сначала надо зайти в ваш профиль. Затем создать репозиторий, для этого нужно перейти во вкладку Repositories и нажать по кнопке New.

Задать имя репозитория. Следующим шагом будет добавление описания (при необходимости). После этого необходимо выбрать режим видимости репозитория (общедоступный или приватный). Далее создаем репозиторий.

Пока проект пустой, но мы можем поместить в него наши файлы с локальной машины.

#### 12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Публичные репозитории на GitHub часто используются для совместного использования программного обеспечения с открытым исходным кодом. Чтобы другие могли свободно использовать, изменять и распространять программное обеспечение, вам нужно будет лицензировать его.

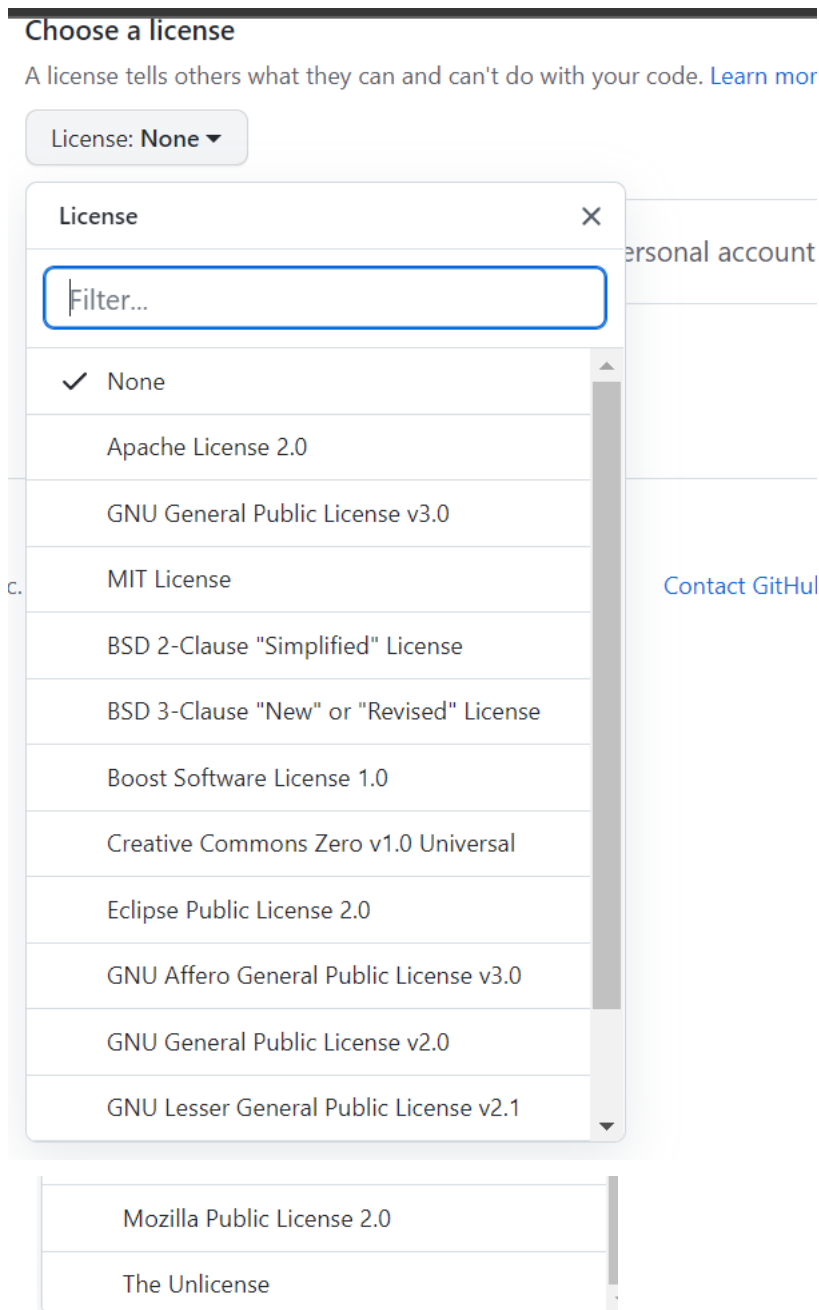


Рисунок 1.2– Лицензии GitHub

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

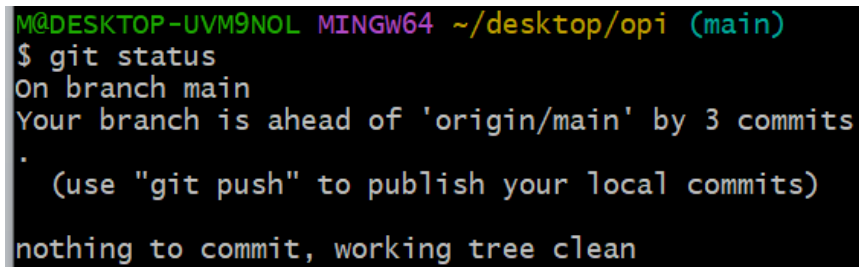
```
git clone https://github.com/obrMaria/OPI_LR_1.git
```

Клонирование репозитория локально хранит последние изменения проекта, позволяя вам разветвляться и вносить свои собственные изменения, не затрагивая сразу чужую работу. Для этого вам нужно будет загрузить Git или другое программное обеспечение, поддерживаемое Git, найти

репозиторий, который вы хотите клонировать, и указать местоположение для сохранения клонированного репозитория.

#### 14. Как проверить состояние локального репозитория Git?

Проверяем статус локального репозитория - “git status”, в ответ получаем “On branch master. nothing to commit, working directory clean”.



```
M@DESKTOP-UVM9NOL MINGW64 ~/desktop/opi (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits
  (use "git push" to publish your local commits)
nothing to commit, working tree clean
```

Рисунок 1.3– проверка состояния локального репозитория

15. Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/ измененного файла под версионный контроль с помощью команды git add ; фиксации (коммита) изменений с помощью команды git commit и отправки изменений на сервер с помощью команды git push ?

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 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`.

Чтобы связать новый проект на GitHub с папкой проекта на компьютере, надо:

создать проект на GitHub (новый репозиторий) и скопировать его URL  
перейти в Терминале в общую папку для проектов  
клонировать проект с GitHub — `git clone URL`

Чтобы локальные изменения синхронизировались с удалённым репозиторием, необходимо их сначала добавить, потом закоммитить, а потом запустить.

`Git add .`

`Git commit -m “название коммита”`

`Git push`

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

GitKraken обладает прекрасным интерфейсом. Он ориентирован на скорость и простоту использования Git. Цель платформы — экономить время на сборку и тестирование кода. Он поддерживает GitHub, Bitbucket и Gitlab, и конечно же позволяет работать с корпоративными репозиториями с исходным кодом.

Возможности:

- Визуальное взаимодействие и подсказки;
- Поддержка нескольких профилей;
- Поддерживает кнопки отмены и повтора функции;
- Быстрый и интуитивно понятный интерфейс поиска;
- Легко адаптируется к рабочей области пользователя, а также поддерживает подмодули и Git-flow;



- Интегрируется с аккаунтами на GitHub или BitBucket;
- Горячие клавиши и многое другое.

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.

GitHub Desktop это совершенно бесплатное приложение с открытым исходным кодом, разработанное GitHub. С его помощью можно взаимодействовать с GitHub (что и не удивительно), а также с другими платформами (включая Bitbucket и GitLab).

Он дублирует функциональность сайта github.com, но при этом работает локально на компьютере разработчика, можно сразу привязать свой аккаунт.

Графический интерфейс позволяет отслеживать историю всех изменений:

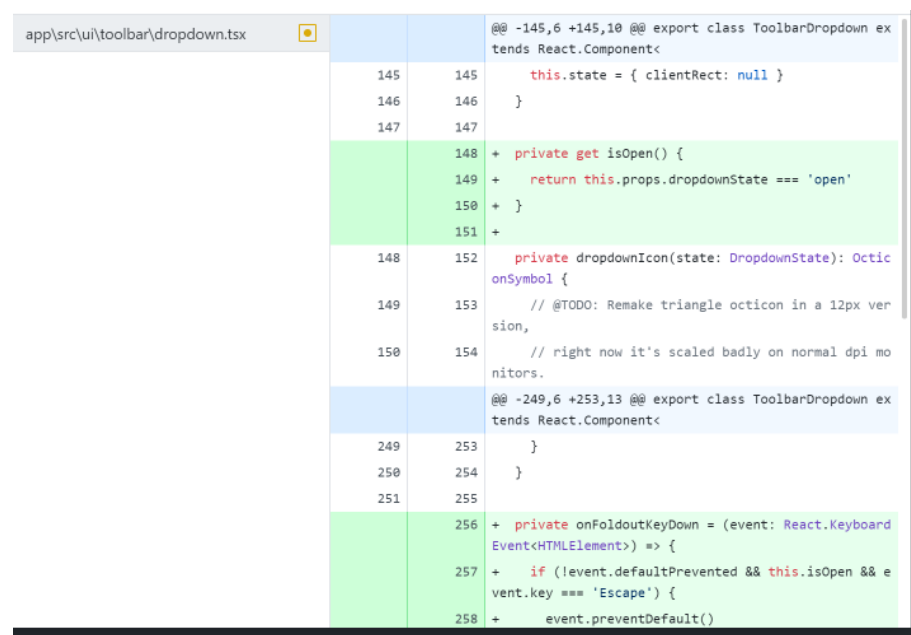


Рисунок 1.4 – История изменений в GitHub Desktop

Можно клонировать/ создать / добавить репозиторий.

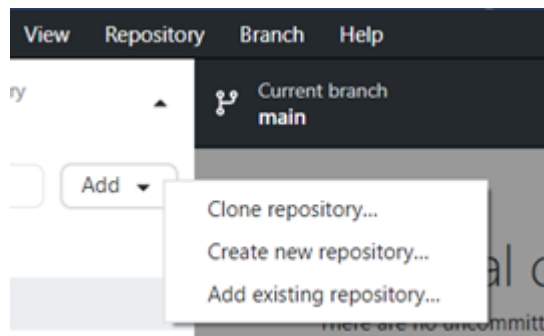


Рисунок 1.5 – Работа с репозиториями в GitHub Desktop

При нажатии на клонирование можно использовать ссылку (URL или выбрать из имеющихся на сайте репозиториях).

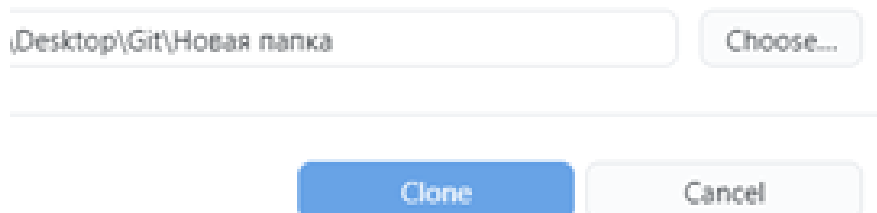


Рисунок 1.6 – Клонирование репозитория в GitHub Desktop

Создание коммита производится с помощью нажатия на кнопку «Commit to main», так же можно добавить комментарий.

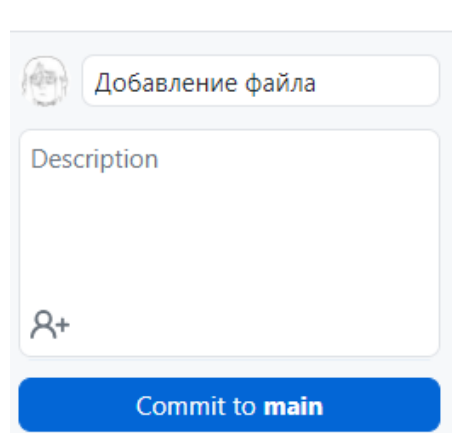


Рисунок 1.7 –коммит в GitHub Desktop

Сразу после этого нажимаем на кнопку «Push origin» и отправляем на сервер.

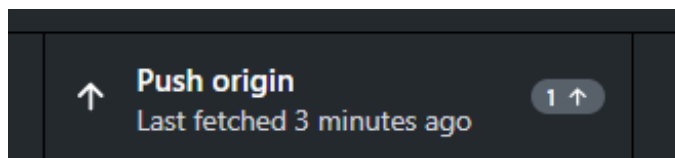
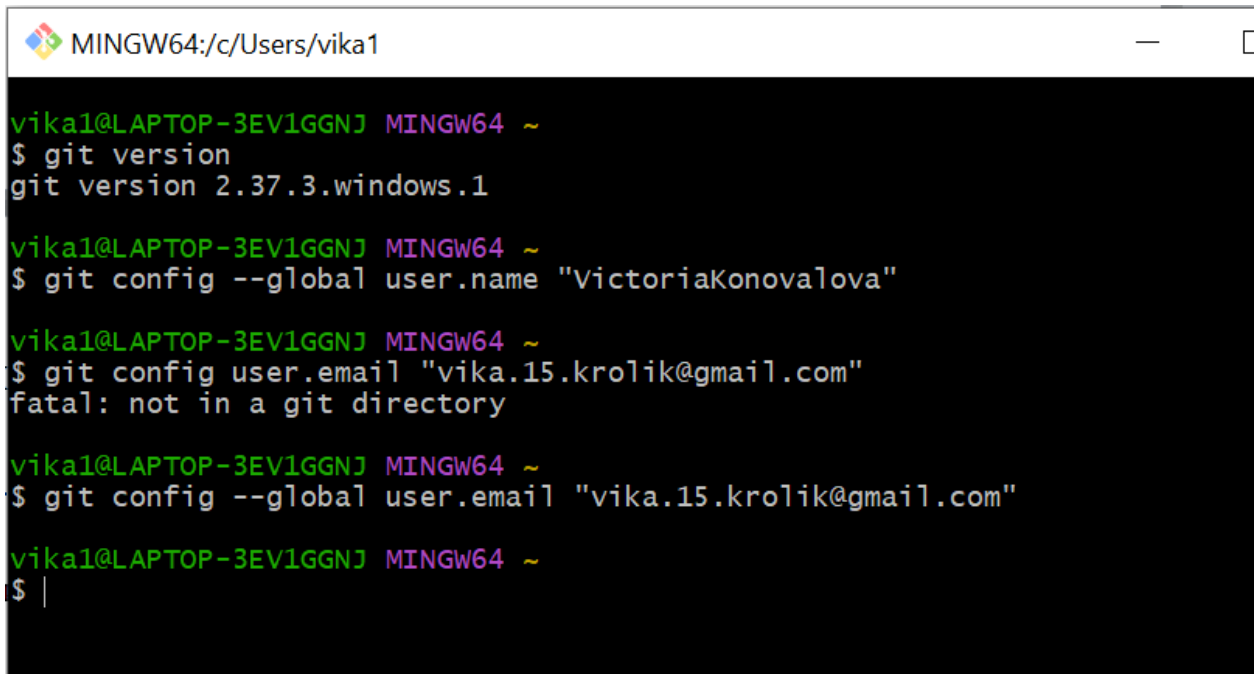


Рисунок 1.8 – Отправка изменений на сервер в GitHub Desktop

Ход работы:

## 1. Установка и настройка Git

A screenshot of a terminal window titled 'MINGW64:/c/Users/vika1'. The terminal shows the following commands and output:

```
vika1@LAPTOP-3EV1GGNJ MINGW64 ~  
$ git version  
git version 2.37.3.windows.1  
  
vika1@LAPTOP-3EV1GGNJ MINGW64 ~  
$ git config --global user.name "VictoriaKonovalova"  
  
vika1@LAPTOP-3EV1GGNJ MINGW64 ~  
$ git config user.email "vika.15.krolik@gmail.com"  
fatal: not in a git directory  
  
vika1@LAPTOP-3EV1GGNJ MINGW64 ~  
$ git config --global user.email "vika.15.krolik@gmail.com"  
  
vika1@LAPTOP-3EV1GGNJ MINGW64 ~  
$ |
```


Рисунок 1.1 – Установка и настройка Git

## 2. Создание репозитория GitHub

(Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add .gitignore).)

---

Owner \*

 VictoriaKonovalova ▾

/


Repository name \*

Lr1 ✓


Great repository names are short and memorable. Need inspiration? How about **solid-robot**?

Description (optional)

---

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

---

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore


Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: VisualStudio ▾


Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▾

This will set  **main** as the default branch. Change the default name in your [settings](#).

---

 You are creating a public repository in your personal account.

---

Create repository

Рисунок 1.2 – Создание нового репозитория

После изучения теории, был создан общедоступный репозиторий на GitHub с использованием лицензии MIT и языка программирования C++

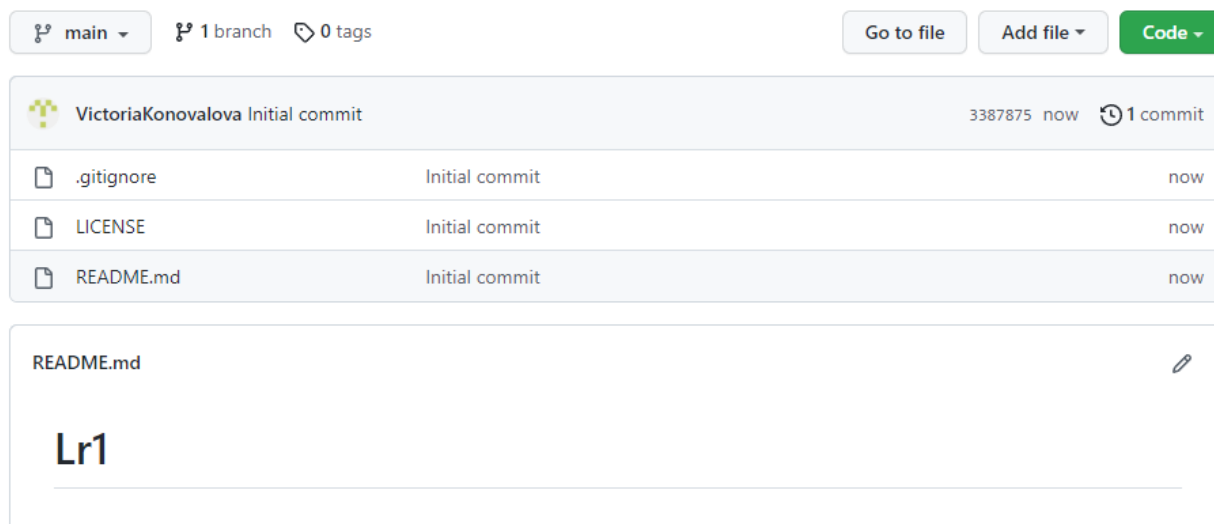


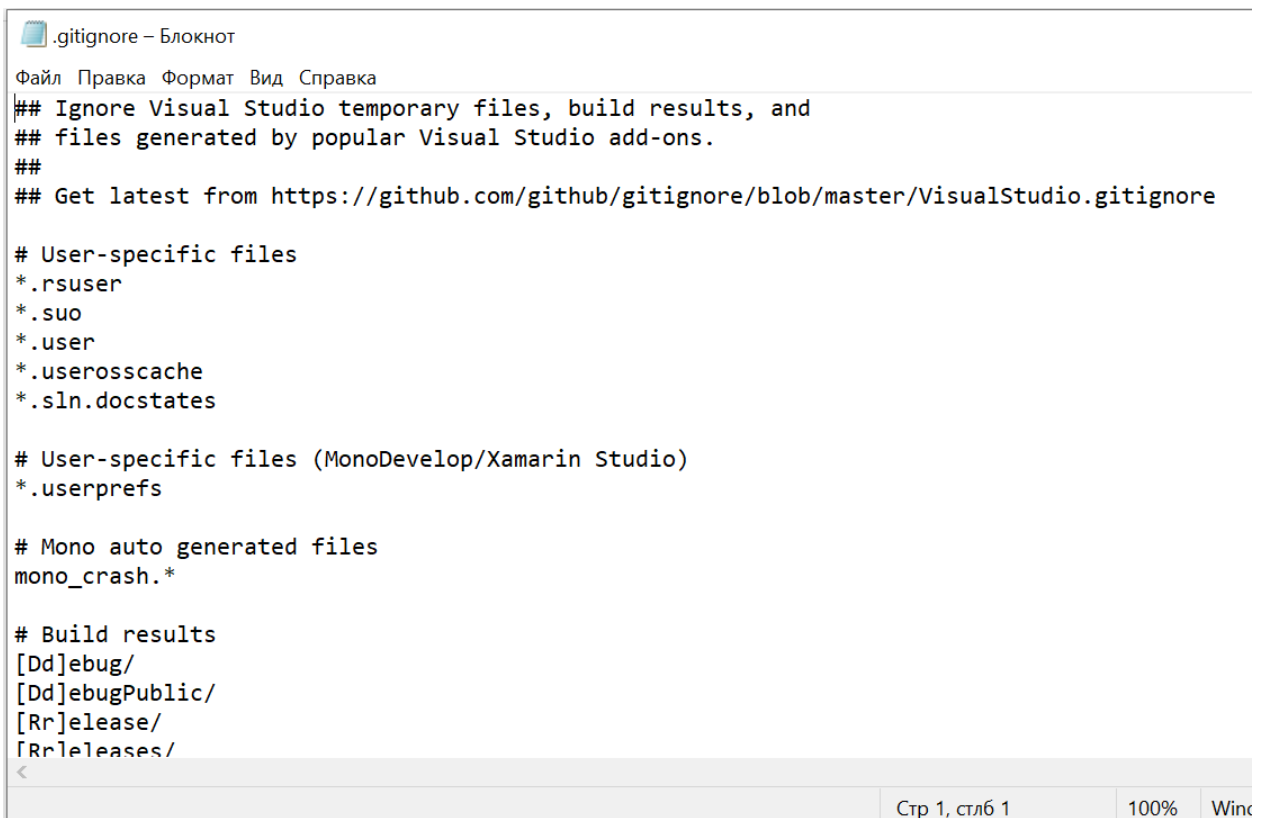
Рисунок 2.1 – Создание репозитория Lr\_1

После этого было выполнено создание локальной копии репозитория на компьютере с помощью команды «git clone».

```
vika1@LAPTOP-3EV1GGNJ MINGW64 ~
$ cd "C:\Users\vika1\OneDrive\Desktop\univer\Основы программной инженерии"
vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии
$ git clone https://github.com/VictoriaKonovalova/Lr1.git
Cloning into 'Lr1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии
$ |
```

Рисунок 2.2 – Клонирование репозитория

Так как при создании репозитория был указан язык, файл .gitignore был автоматически заполнен необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.



The screenshot shows a Notepad window titled ".gitignore – Блокнот". The menu bar includes "Файл", "Правка", "Формат", "Вид", and "Справка". The text content of the file is as follows:

```
## Ignore Visual Studio temporary files, build results, and
## files generated by popular Visual Studio add-ons.
##
## Get latest from https://github.com/github/gitignore/blob/master/VisualStudio.gitignore

# User-specific files
*.rsuser
*.suo
*.user
*.useroscachе
*.sln.docstates

# User-specific files (MonoDevelop/Xamarin Studio)
*.userprefs

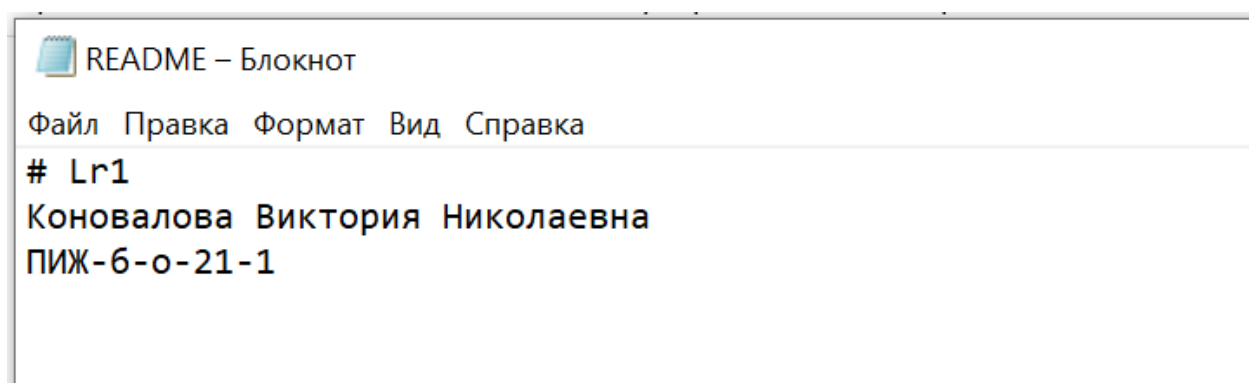
# Mono auto generated files
mono_crash.*

# Build results
[Dd]ebug/
[Dd]ebugPublic/
[Rr]elease/
[Rr]eleases/
```

The status bar at the bottom right shows "Стр 1, стлб 1", "100%", and "Win".

Рисунок 1.6 – Файл «.gitignore»

Далее файл README.md был дополнен информацией об имени студента и группе.



The screenshot shows a Notepad window titled "README – Блокнот". The menu bar includes "Файл", "Правка", "Формат", "Вид", and "Справка". The text content of the file is as follows:

```
# Lr1
Коновалова Виктория Николаевна
ПИЖ-6-о-21-1
```

Рисунок 1.7 – Изменения в файле README.md.

```

vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии/Lr1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

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

```

Рисунок 1.8 – Проверка статуса репозитория

```

vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы програм
/Lr1 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

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

vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы програм
/Lr1 (main)
$ git commit -m "КОММИТ 2 RM"
[main debfe44] КОММИТ 2 RM
1 file changed, 1 insertion(+), 1 deletion(-)

vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы програм
/Lr1 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы програм
/Lr1 (main)
$ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 660 bytes | 165.00 KiB/s, done.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/VictoriaKonovalova/Lr1.git
3387875..debfe44  main -> main

```

Рисунок 1.9 – Отправка изменения в удаленный репозиторий

Была написана небольшая программа на языке программирования C++. Также были зафиксированны изменения при написании программы в локальном репозитории с помощью команд «git add .», «git commit», «git push».



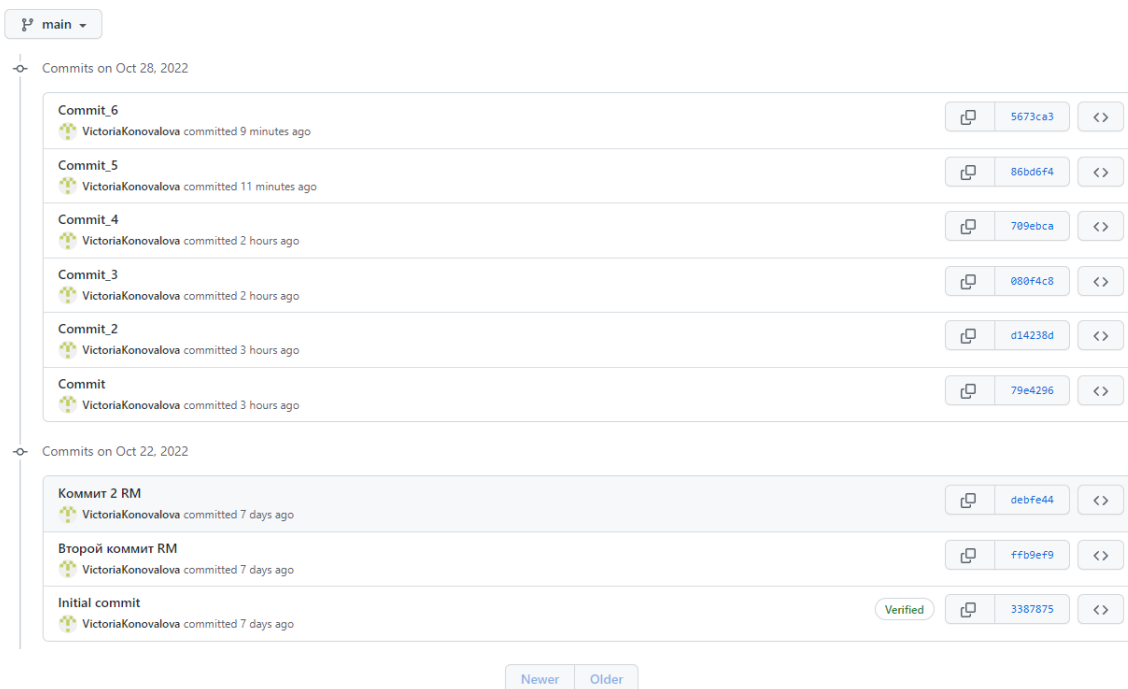


Рисунок 1.10 – Зафиксированные изменения

Далее был создан файл README\_new.txt, внесен в папку с локальным репозиторием и зафиксирован в коммите.

```
vika1@LAPTOP-3EVLGGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии/Lr1 (main)
$ echo "новый README файл" > README_new.txt

vika1@LAPTOP-3EVLGGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии/Lr1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  README_new.txt

nothing added to commit but untracked files present (use "git add" to track)

vika1@LAPTOP-3EVLGGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии/Lr1 (main)
$ git add .
warning: in the working copy of 'README_new.txt', LF will be replaced by CRLF the next time Git touches it

vika1@LAPTOP-3EVLGGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии/Lr1 (main)
$ 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)
  new file:   README_new.txt
```

Рисунок 1.11 – Добавление файла README\_new.txt

После этого с помощью команды «git push» локальный репозиторий был отправлен на удаленный.

```
vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии/Lr1 (main)
$ git commit -m "новый README файл"
[main 9bae026] новый README файл
1 file changed, 1 insertion(+)
create mode 100644 README_new.txt

vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии/Lr1 (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 361 bytes | 180.00 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/VictoriaKonovalova/Lr1.git
5673ca3..9bae026  main -> main

vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии/Lr1 (main)
$
```

Рисунок 1.12 – распространение изменений в исходном репозитории на GitHub.

После всех сделанных действий и отправки локального репозитория в удаленный репозиторий GitHub мы можем наблюдать следующие изменения:

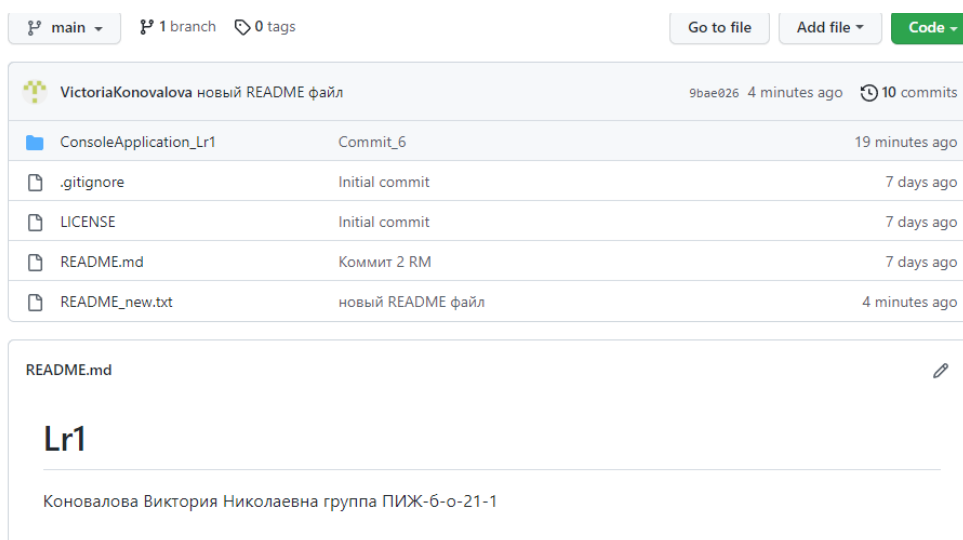


Рисунок 1.13 – Страница на GitHub после внесенных изменений

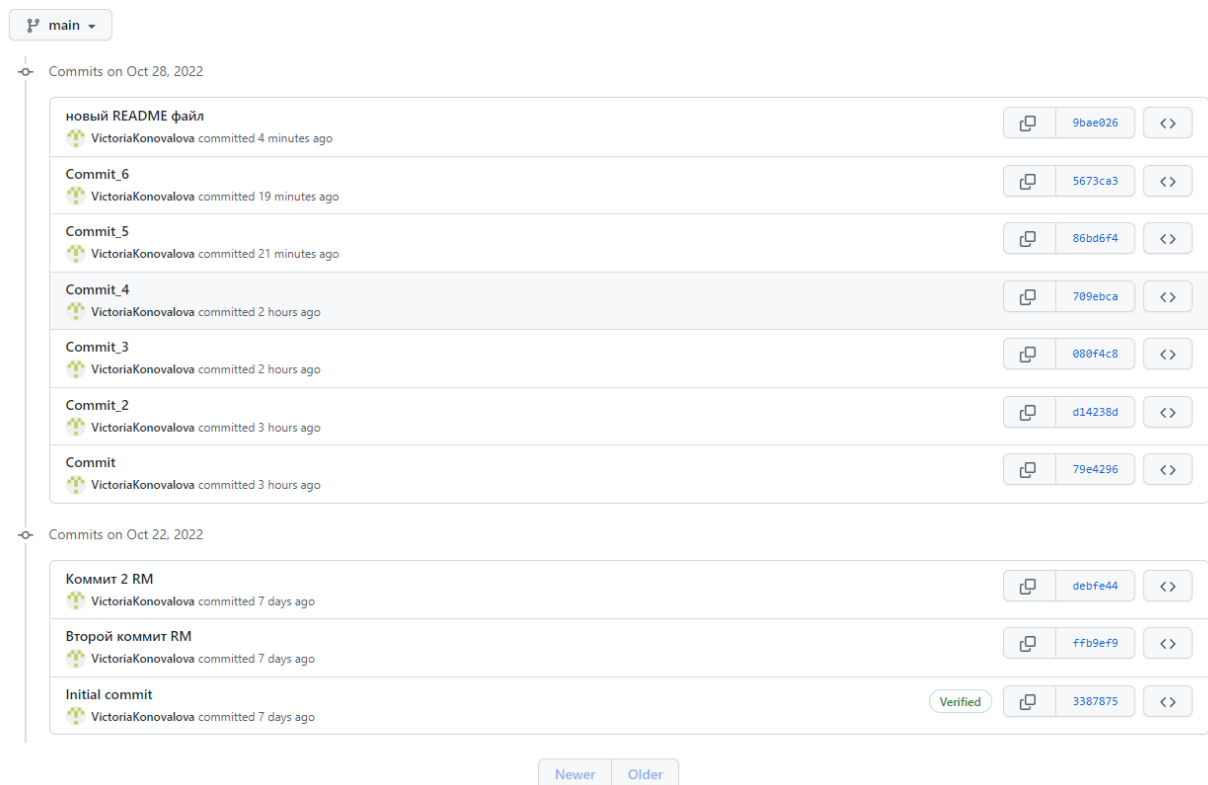


Рисунок 1.14 – Все коммиты, просмотренные в GitHub

```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  "\320\232\320\276\320\275\320\276\320\262\320\260\320\273\320\276\320\26
2\320\260_\320\276\321\202\321\207\321\221\321\202_1.docx"

nothing added to commit but untracked files present (use "git add" to track)

vika1@LAPTOP-3EVI6GNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии/Lr1 (main)
$ git add .

vika1@LAPTOP-3EVI6GNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии/Lr1 (main)
$ 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)
    new file:   "\320\232\320\276\320\275\320\276\320\262\320\260\320\273\320\276\320\262\320\260_\320\276\321\202\321\207\321\221\321\202_1.docx"

vika1@LAPTOP-3EVI6GNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии/Lr1 (main)
$ git commit -m "Отчёт"
[main c491ded] Отчёт
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 "\320\232\320\276\320\275\320\276\320\262\320\260\320\273\320\276\320\262\320\260_\320\276\321\202\321\207\321\221\321\202_1.docx"

vika1@LAPTOP-3EVI6GNJ MINGW64 ~/OneDrive/Desktop/univer/Основы программной инженерии/Lr1 (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.15 MiB | 67.00 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/Victoriakonovalova/Lr1.git
  9bae026..c491ded main -> main
```

Рисунок 1.10 – Добавление начального отчета в формате .doc

**Выводы:** в ходе лабораторной работы были изучены основы работы с сервисом GitHub, а также базовые команды системы контроля версий Git.