

**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

Халимендик Я. Д. « 3 » сентября 2022г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2022

| | |
|---|----|
| «Исследование основных возможностей Git и GitHub» | 0 |
| 1. Теоретическая часть..... | 1 |
| 2. Практическая часть..... | 13 |

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

Ход работы:

1. Теоретическая часть

1. Что такое СКВ и каково ее назначение?

СКВ (система контроля версий) – это хранилище, которое позволяет фиксировать содержание файла и его изменения на разных этапах работы с ним, а также свободно возвращаться к более старым версиям, что поможет при, например, возникновении ошибки.

2. В чем недостатки локальных и централизованных СКВ?

Локальные сети значительно усложняют пути взаимодействия между несколькими пользователями. Эту проблему решают централизованные СКВ, где есть один сервер, предоставляющий доступ к файлам нескольким людям, но в случае возникновения проблем с сервером, все клиенты не смогут продолжать работу.

3. К какой СКВ относится Git?

Git относится к распределенным СКВ. РСКВ решает проблему одного сервера ЦСКВ тем, что копирует полностью репозиторий на компьютер каждого пользователя.

4. В чем концептуальное отличие Git от других СКВ?

В Git во время сохранения проекта программа запоминает состояние всех файлов (при отсутствии изменений дается ссылка на исходный файл), в

отличие от других СКВ, которые фиксируют только изменения отдельных файлов.

5. Как обеспечивается целостность хранимых данных в Git?

Для обеспечения целостности файлов и их структуры используется хеш-сумма.

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

При работе с файлом, когда создается новый вариант этого файла, но еще не сохраненный, он находится в состоянии измененном. Когда работа закончена, и файл готов к перемещению в следующий коммит, его состояние называется подготовленным. Зафиксированный файл – файл с сохраненными изменениями.

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?

Репозиторий можно сравнить с ячейкой, в которой пользователь хранит проект. Помимо кода он может содержать различные медиа-файлы, инструкции и многое другое. По содержанию репозитории никак не ограничены. Это может быть, как полноценный проект, так и фрагмент кода, шаблон, набор примеров.

9. Укажите основные этапы модели работы с GitHub.

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

10. Как осуществляется первоначальная настройка Git после установки?

Необходимо 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:

```
C:\Users\антон>git version
git version 2.37.3.windows.1

C:\Users\антон>git config --global user.name Yana-Kh

C:\Users\антон>git config --global user.email ynakh2017@mail.ru
```

Рисунок 1.1 – Начальная настройка Git

11. Опишите этапы создания репозитория в GitHub.

Для начала в правом верхнем углу необходимо найти «+» и в появившемся меню выбрать «New repository»

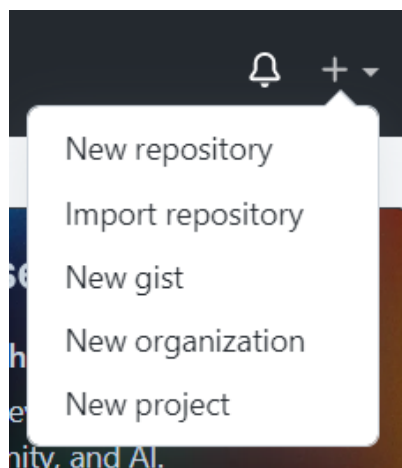
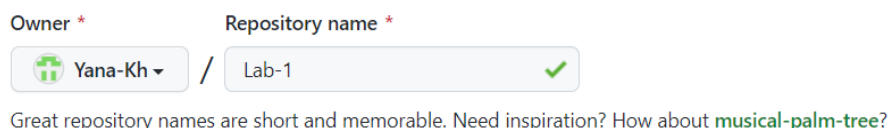


Рисунок 1.2 – Всплывающее меню, при нажатии на «+»

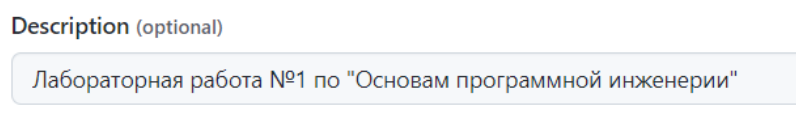
Затем необходимо придумать имя, описывающее Ваш репозиторий. Относительно всего GitHub оно может быть неуникальным, но повторов среди репозитория в Вашем аккаунте быть не должно.



The screenshot shows the GitHub repository creation interface. The 'Owner' field is set to 'Yana-Kh' with a dropdown arrow. The 'Repository name' field is set to 'Lab-1' and has a green checkmark icon to its right. Below the fields, there is a hint: 'Great repository names are short and memorable. Need inspiration? How about musical-palm-tree?'.

Рисунок 1.3 – Заполнение имени репозитория

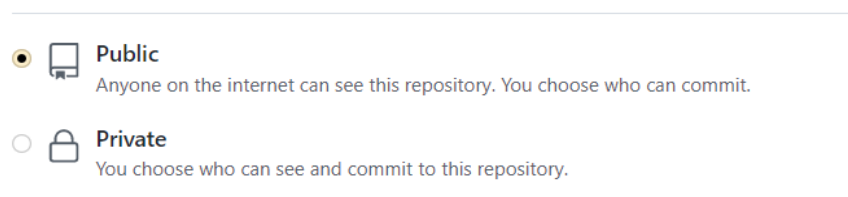
Следующим шагом будет добавление описания (при необходимости).



The screenshot shows the 'Description (optional)' field in the GitHub repository creation interface. The text 'Лабораторная работа №1 по "Основам программной инженерии"' is entered into the text area.

Рисунок 1.4 – Заполнение описания репозитория

После этого необходимо выбрать режим видимости репозитория (общедоступный или приватный).



The screenshot shows the visibility options in the GitHub repository creation interface. The 'Public' option is selected with a radio button and a lock icon. The description below it says: 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option is unselected with a radio button and a lock icon. The description below it says: 'You choose who can see and commit to this repository.'

Рисунок 1.5 – Выбор режима видимости

В конце нажимаем «Create repository». Готово.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Лицензия на программное обеспечение сообщает другим пользователям, что они могут и чего не могут делать с вашим исходным кодом. GitHub предоставляет множество различных лицензий, а также сервис помогающий с подбором и фильтром репозитория по данному критерию.

Вот полный список предоставляемых лицензий:

License, Academic Free License v3.0, Apache license 2.0, Artistic license 2.0, Boost Software License 1.0, BSD 2-clause "Simplified" license, BSD 3-clause "New" or "Revised" license, BSD 3-clause Clear license, Creative Commons license family, Creative Commons Zero v1.0 Universal, Creative Commons Attribution 4.0, Creative Commons Attribution Share Alike 4.0, Do What The F*ck You Want To Public License, Educational Community License v2.0, Eclipse Public License 1.0, Eclipse Public License 2.0, European Union Public License 1.1, GNU Affero General Public License v3.0, GNU General Public License family, GNU General Public License v2.0, GNU General Public License v3.0, GNU Lesser General Public License family, GNU Lesser General Public License v2.1, GNU Lesser General Public License v3.0, ISC, LaTeX Project Public, License v1.3c, Microsoft Public License, MIT, Mozilla Public License 2.0, Open Software License 3.0, PostgreSQL License, SIL Open Font License 1.1, University of Illinois/NCSA Open Source License, The Unlicense, zlib License

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

Клонирование репозитория позволяет свободно экспериментировать с изменениями, не затрагивая исходный проект.

Для того чтобы осуществить это действие необходимо открыть репозиторий, открыть вкладку код и найти в конце строки с адресом кнопку для копирования.

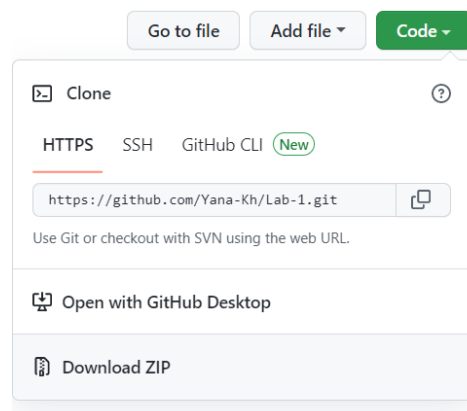


Рисунок 1.6 – Клонирование репозитория

Затем откройте командную строку, перейдите в нужный каталог и запишите «git clone» + ссылка и Enter. Готово

```
C:\Users\антон>cd C:\Users\антон\Desktop\Git  
  
C:\Users\антон\Desktop\Git>git clone https://github.com/Yana-Kh/Lab-1.git  
Cloning into 'Lab-1'...  
remote: Enumerating objects: 3, done.  
remote: Counting objects: 100% (3/3), done.  
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
Receiving objects: 100% (3/3), done.  
  
C:\Users\антон\Desktop\Git>
```

Рисунок 1.7 – Создание локальной копии репозитория с помощью командной строки

14. Как проверить состояние локального репозитория Git?

Для этого необходимо с помощью командной строки перейти в интересующий каталог и использовать команду «git status»

```
C:\Users\антон\Desktop\Git>cd C:\Users\антон\Desktop\Git\Lab-1  
  
C:\Users\антон\Desktop\Git\Lab-1>git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
nothing to commit, working tree clean
```

Рисунок 1.8 – Проверка статуса локальной копии репозитория с помощью командной строки

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?

Количество коммитов обнулиться, а так как работа проводилась на локальном репозитории Вы будете иметь дело с последними версиями файлов.

```
C:\Users\антон\Desktop\Git\Lab-1>git push
info: please complete authentication in your browser...
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), 13.47 KiB | 1.22 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Yana-Kh/Lab-1.git
   b442a91..698665f  main -> main

C:\Users\антон\Desktop\Git\Lab-1>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\антон\Desktop\Git\Lab-1>_
```

Рисунок 1.9 – Проверка статуса локальной копии репозитория после отправки изменений на сервер

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`.

После клонирования репозитория на свое устройство с помощью «`git clone`» для просмотра списка настроенных удалённых репозитория, можно использовать команду «`git remote`». Она выведет названия доступных удалённых репозитория. Если вы клонировали репозиторий, то увидите, как минимум `origin` — имя по умолчанию, которое Git даёт серверу, с которого производилось клонирование.

Для получения изменений из удалённого репозитория используется команда «`git fetch [remote-name]`». Она связывается с указанным удалённым проектом и забирает все те данные проекта, которых у вас ещё нет.

Отправка изменений происходит следующим образом «`git push <remote-name> <branch-name>`».

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

Один из сервисов, работающий с Git – GitLab. При сравнении с GitHub первое что бросается в глаза – платная подписка. GitHub является бесплатным сервисом, что делает его доступным для каждого. Отличной и достаточно интересной функцией GitLab является «планирование», также GitLab предлагает встроенную систему тестирования, отслеживания ошибок и мониторинга. Вероятно, GitLab будет более удобна для работы в команде, так как основной фишкой сервиса является сопровождение на всех этапах разработки, возможность назначать роли.

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.

Одним из графических интерфейсов для работы с Git является GitHub Desktop. Он дублирует функциональность сайта github.com, но при этом работает локально на компьютере разработчика, можно сразу привязать свой аккаунт.

Вид начального окна представлен ниже:

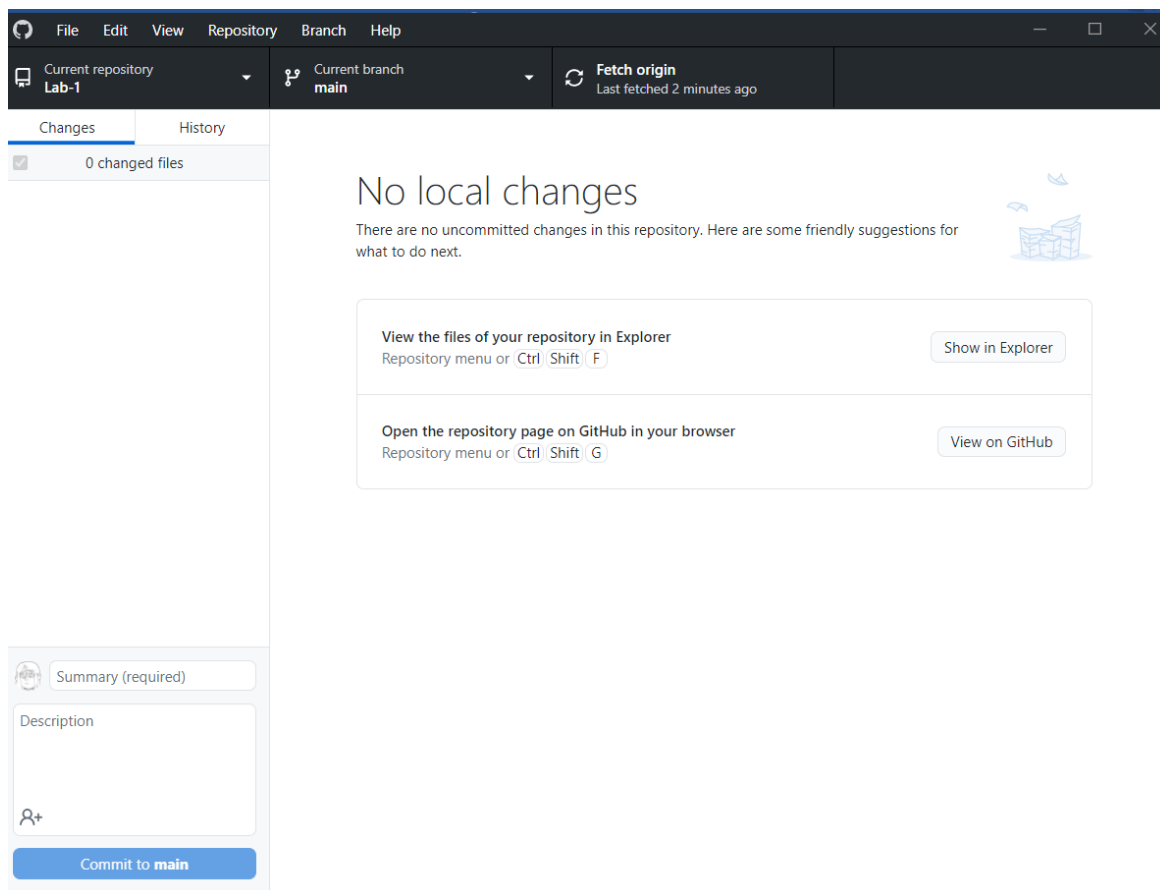


Рисунок 1.10 – Начальное окно при открытии GitHub Desktop

Также графический интерфейс позволяет отслеживать историю всех изменений:

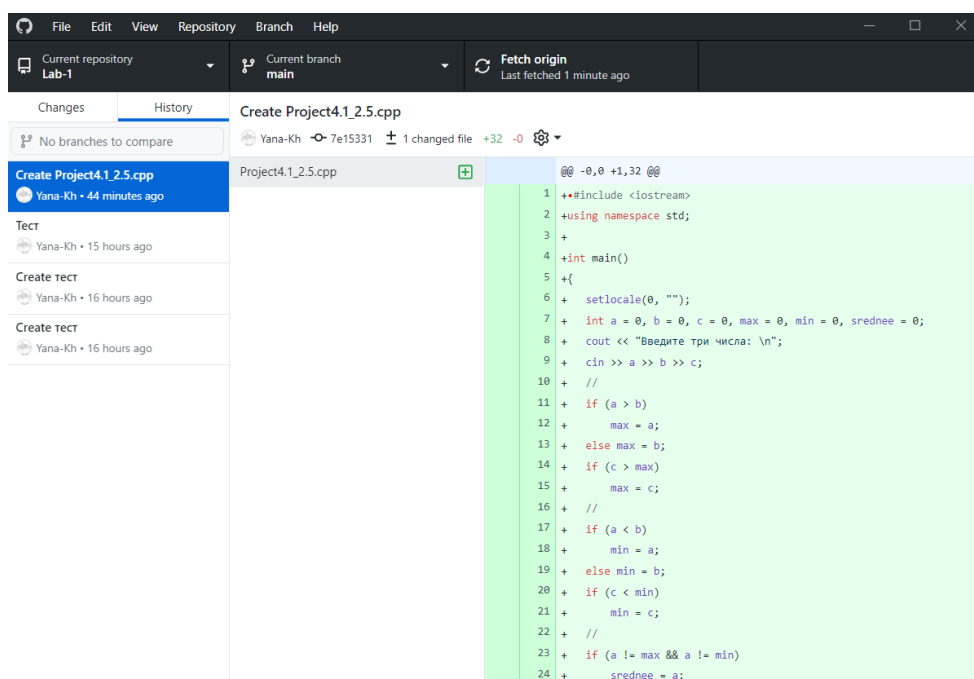


Рисунок 1.11 – История изменений в GitHub Desktop

Есть возможность клонировать репозиторий, создать новый или добавить уже имеющуюся на вашем устройстве локальную копию.

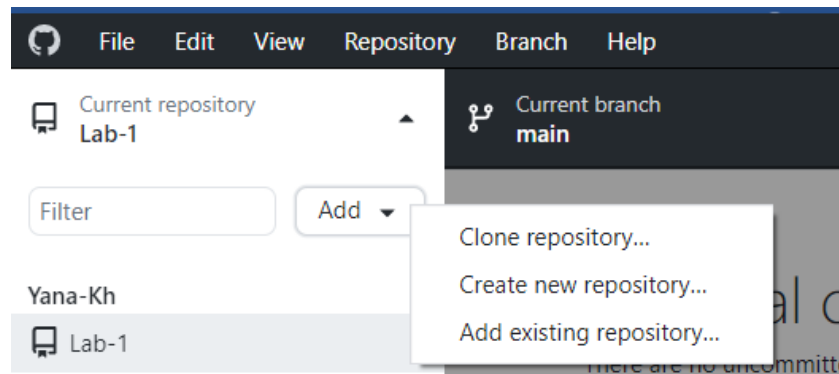


Рисунок 1.12 – Работа с репозиториями в GitHub Desktop

При нажатии на клонирование можно использовать ссылку (URL), а можно выбрать из имеющихся на сайте репозиториях, закрепленных за аккаунтом, в который был выполнен вход.

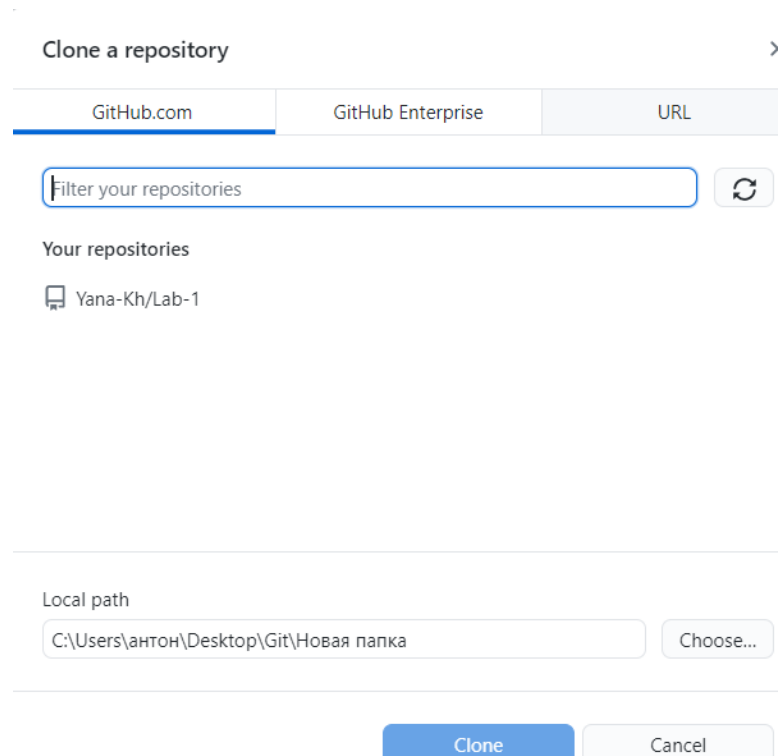


Рисунок 1.13 – Клонирование репозитория в GitHub Desktop

Есть возможность также создать новую ветку, при нажатии на «New branch», будет предложено заполнить имя этой ветки.

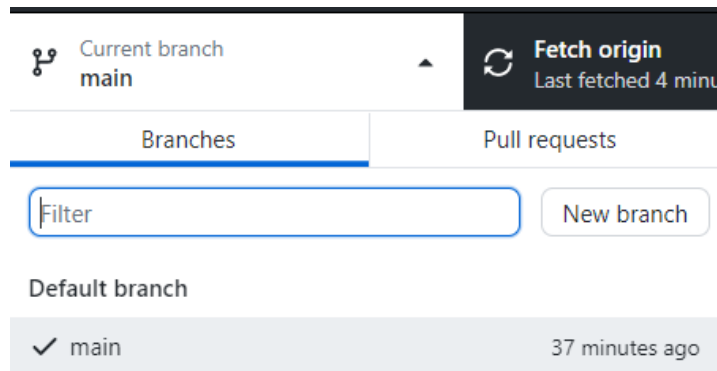


Рисунок 1.14 – Добавление новой ветки в GitHub Desktop

После добавление или модификации файла в локальной копии репозитория, GitHub Desktop предложит просмотреть изменения, что можно сравнить с командой «git status».

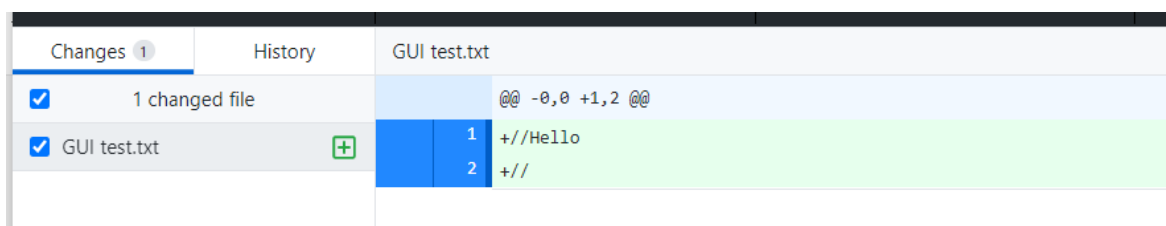


Рисунок 1.15 – Просмотр изменений в GitHub Desktop

Создание коммита, которое при использовании командной строки осуществляется с помощью «git commit», сводится к добавлению комментария и кнопке «Commit to main»

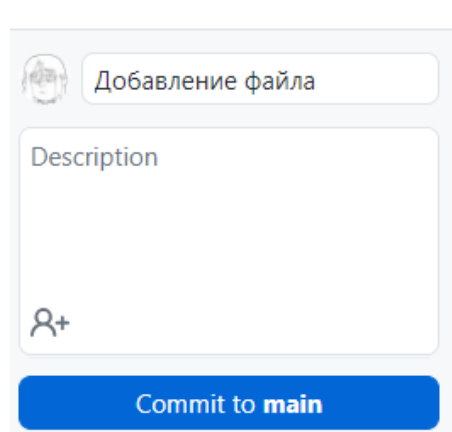


Рисунок 1.16 – Создание коммита в GitHub Desktop

Сразу после этого можно увидеть, что сохранились фиксации, еще не отправленные сервер. Нажатие кнопки «Push origin» заменяет команду «git push» в командной строке.

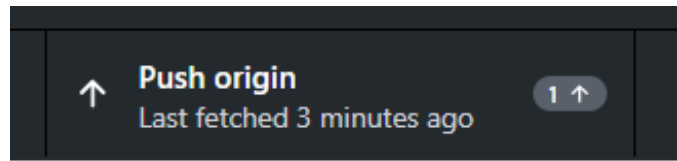


Рисунок 1.17 – Отправка изменений на сервер в GitHub Desktop

Результат проделанной работы тут же отображается на сайте GitHub.

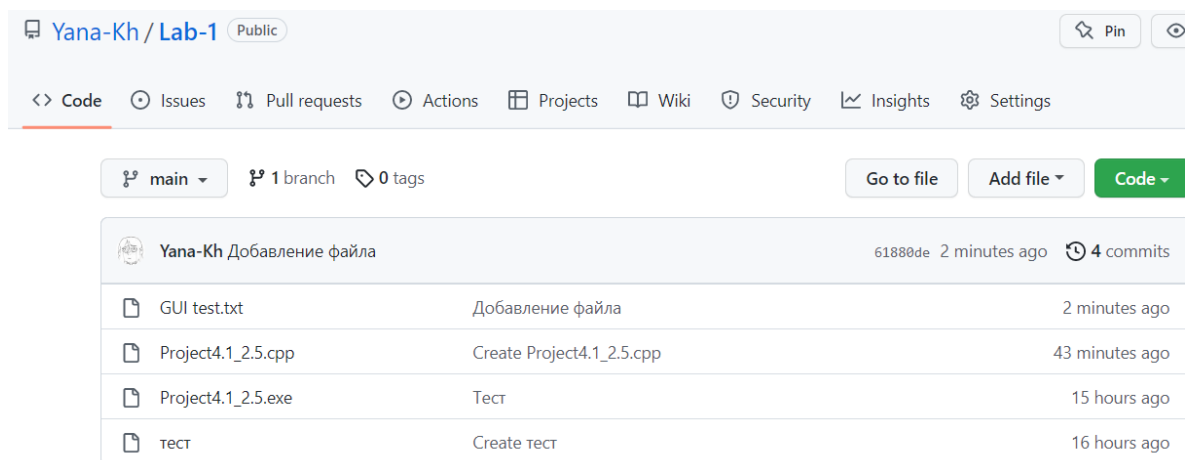


Рисунок 1.18 – Изменения, которые мы видим на сайте GitHub

2. Практическая часть

После изучения теории, был создан общедоступный репозиторий на GitHub с использованием лицензии MIT и языка программирования C++ (<https://github.com/Yana-Kh/Lab-1-OPJ.git>)

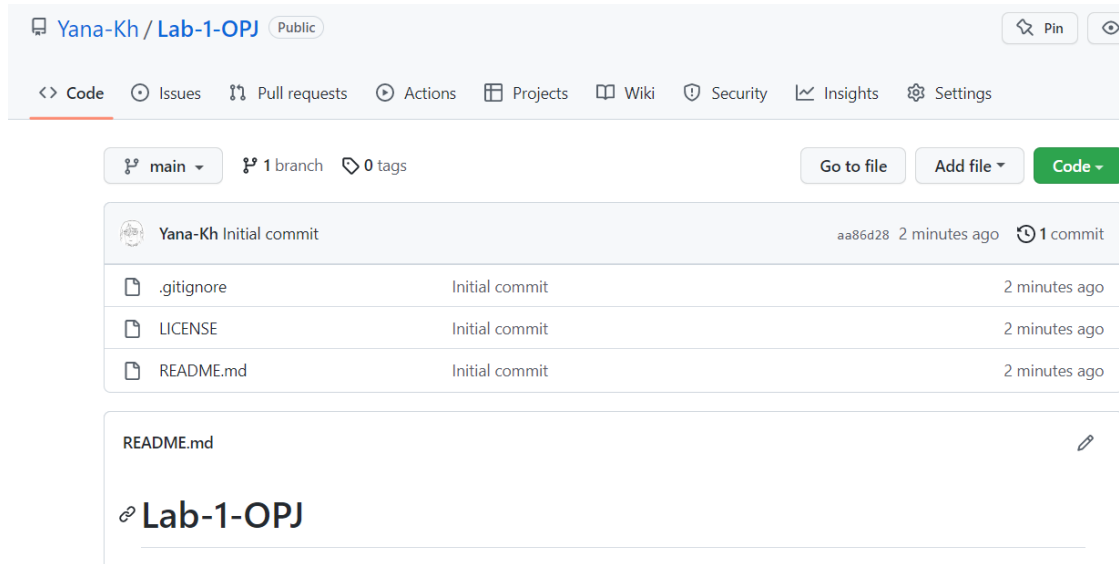


Рисунок 2.1 – Создание репозитория « Lab-1-OPJ»

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

```
Выбрать C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1889]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\антон>cd C:\Users\антон\Desktop\Git

C:\Users\антон\Desktop\Git>git clone https://github.com/Yana-Kh/Lab-1-OPJ.git
Cloning into 'Lab-1-OPJ'...
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.

C:\Users\антон\Desktop\Git>
```

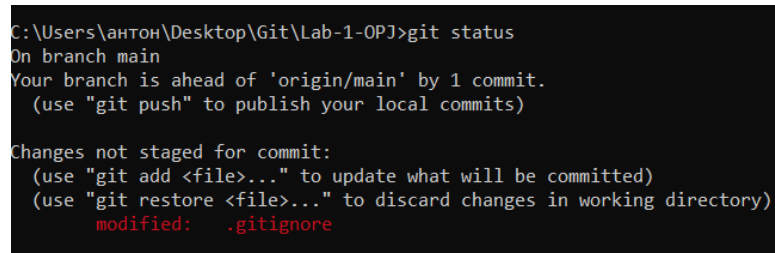
Рисунок 2.2 – Клонирование репозитория

Следующим шагом было дополнение файла .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки, но так как при создании репозитория был указан язык, файла .gitignore был автоматически заполнен. Было принято решение

добавить еще пару строк, для получения практических навыков работы с файлом.

```
#Дополнение .gitignore
*.xlsx #исключить файлы с таким расширением
! *.png # НЕ исключать файлы с таким расширением
```

Рисунок 2.3 – Изменения внесенные в файл . gitignore



```
C:\Users\антон\Desktop\Git\Lab-1-ОПЈ>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

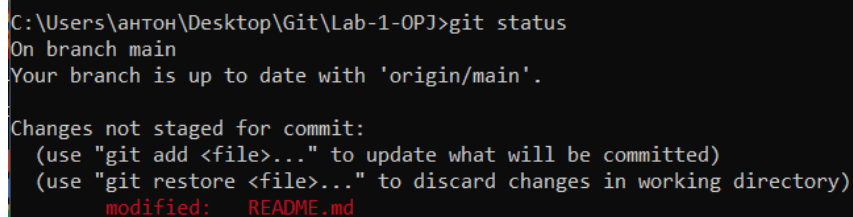
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore
```

Рисунок 2.4 – Проверка статуса репозитория

Далее файл README.md был дополнен информацией об имени студента и номере академической группы.

```
# Lab-1-ОПЈ
# Халимендик Яна Денисовна
# группа ПИЖ-б-о-21-1
```

Рисунок 2.5 – Изменения в файле README.md



```
C:\Users\антон\Desktop\Git\Lab-1-ОПЈ>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
```

Рисунок 2.6 – Проверка статуса репозитория

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

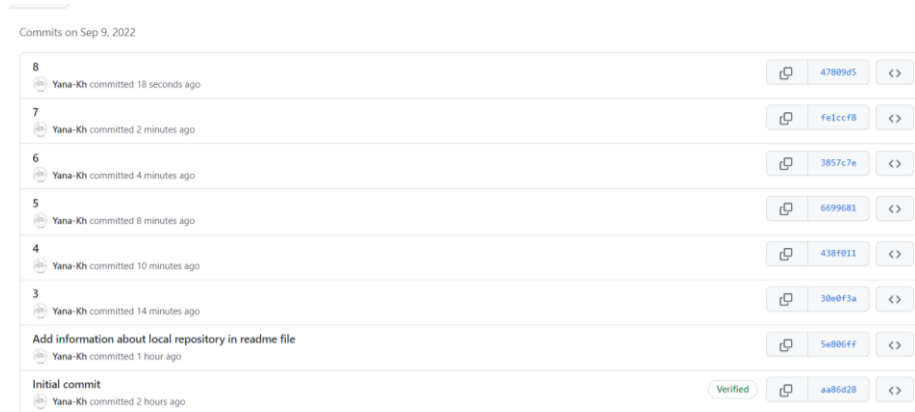


Рисунок 2.7 – Зафиксированные изменения

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

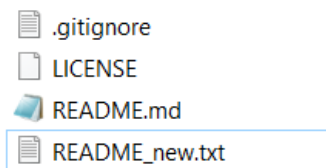


Рисунок 2.8 – Добавление файла README_new.txt

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

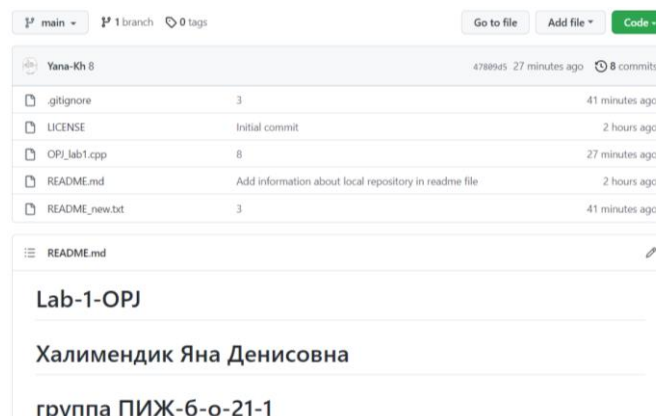


Рисунок 2.9 – Страница на GitHub после внесенных изменений

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