

**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

Кучеренко С.Ю. « 24 » 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/kucherenkosveta/LR_1.git)

Ответы на контрольные вопросы

1. Что такое СКВ и каково ее назначение?

Системы контроля версий — это программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени. В свете усложнения сред разработки они помогают командам разработчиков работать быстрее и эффективнее.

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?

Git относится к распределенным системам, поэтому не зависит от центрального сервера, где хранятся файлы. Git сохраняет данные в локальном репозитории.

4. В чем концептуальное отличие Git от других СКВ?

Главное отличие Git'a от любых других СКВ - 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?

Локальные, которые хранятся на нашей машине, в рабочей папке проекта. Это та самая скрытая папка `git`, и удаленные, которые хранятся 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?

Первое что нужно сделать, ввести свое имя пользователя и адрес электронной почты, можно также проверить версию установленного продукта – это и будет первоначальной настройкой.

```
svetik@MacBook-Air-Svetik LR_1 % git config --global user.name  
kucherenkosveta  
svetik@MacBook-Air-Svetik LR_1 % git config --global user.email  
kucherenko55sss55@gmail.com
```

Рисунок 1.1– первоначальная настройка Git

11. Опишите этапы создания репозитория в GitHub.

Для создания репозитория нужно зайти в профиль, затем создать репозиторий, для этого нужно перейти во вкладку Repositories и нажать по кнопке New и задать имя репозитория. После этого необходимо выбрать режим видимости репозитория (общедоступный или приватный). Далее создаем репозиторий.

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.

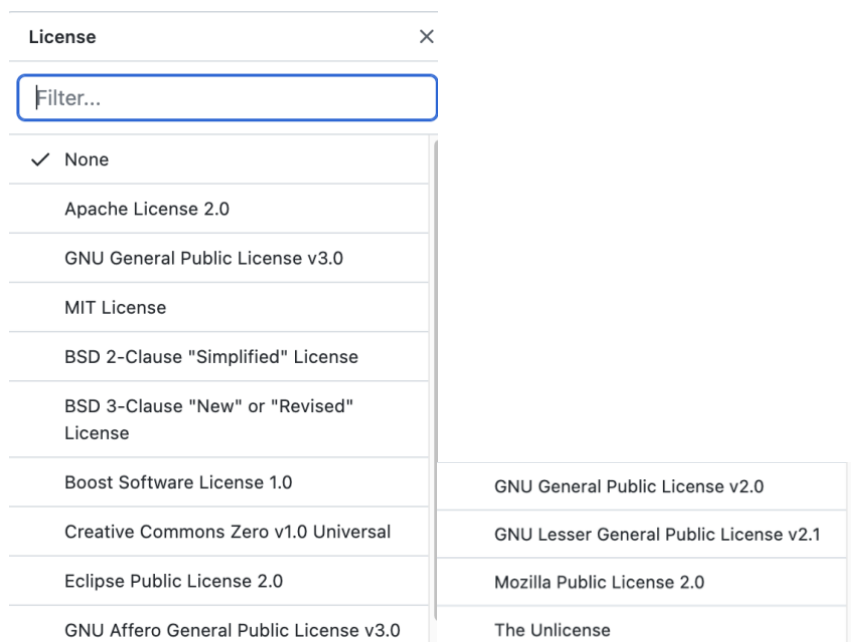


Рисунок 1.2— Доступные лицензии на GitHub

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

Клонирование осуществляется с помощью команды `git clone https://github.com/kucherenkosveta/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 нужно загрузить Git или другое программное обеспечение, поддерживаемое Git, найти репозиторий, который хотим клонировать, и указать местоположение для сохранения клонированного репозитория.

14. Как проверить состояние локального репозитория Git?

Проверить состояние локального репозитория можно при помощи команды в терминале `git status`

```
[svetik@MacBook-Air-Svetik LR_1 % git status
On branch main
Your branch is ahead of 'origin/main' by 7 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`. 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 — это кроссплатформенный, элегантный и высокоэффективный клиент Git для Linux. Он работает в Unix-подобных системах, таких как Linux и Mac OS X, а также в Windows. Он предназначен для повышения производительности пользователя Git с помощью таких функций, как:

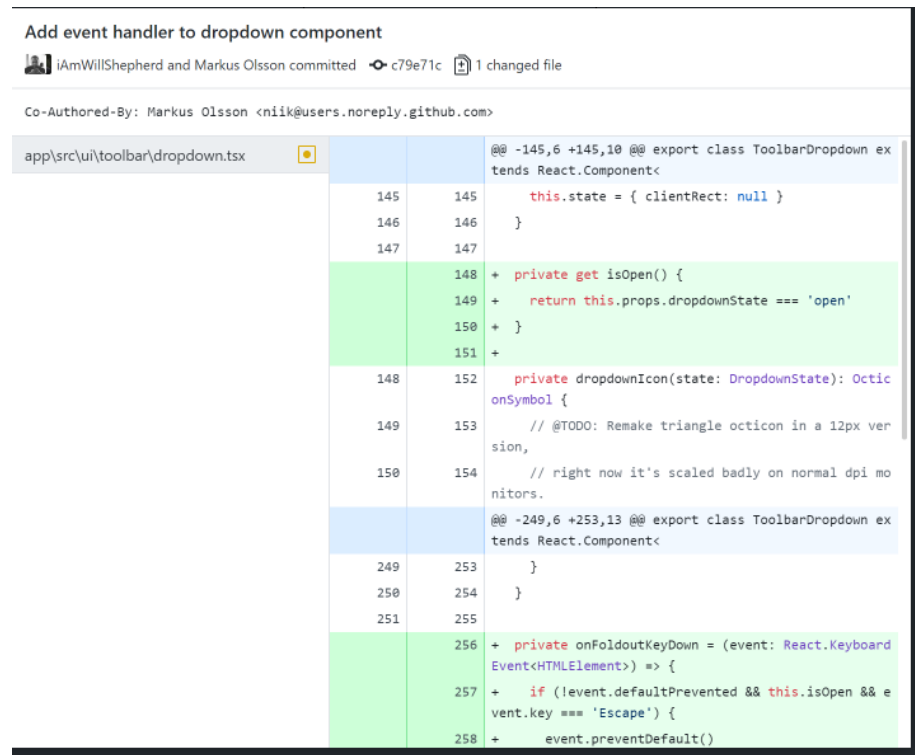
1. Визуальное взаимодействие и подсказки
2. 100% автономный
3. Поддерживает несколько профилей
4. Поддерживает функции отмены и повтора одним щелчком мыши.
5. Встроенный инструмент слияния
6. Быстрый и интуитивно понятный инструмент поиска
7. Легко адаптируется к рабочему пространству пользователя, а также поддерживает подмодули и Gitflow.
8. Интегрируется с учетной записью пользователя GitHub или Bitbucket.
9. Сочетания клавиш 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.

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:



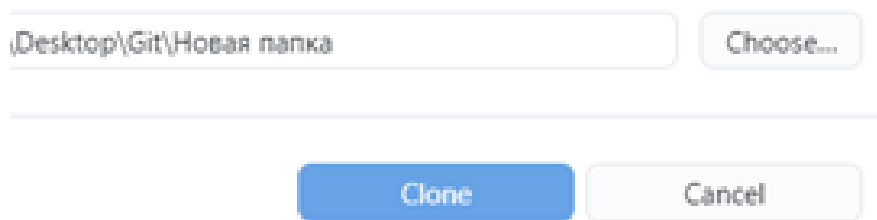


Рисунок 1.6 – Клонирование репозитория в GitHub Desktop

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

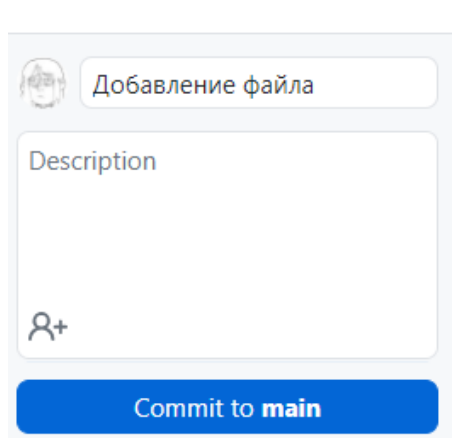


Рисунок 1.7 –коммит в GitHub Desktop

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

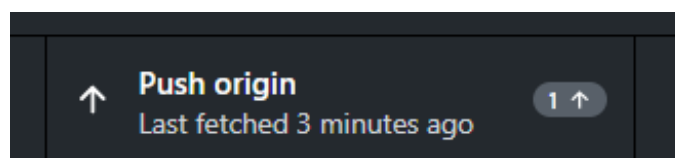
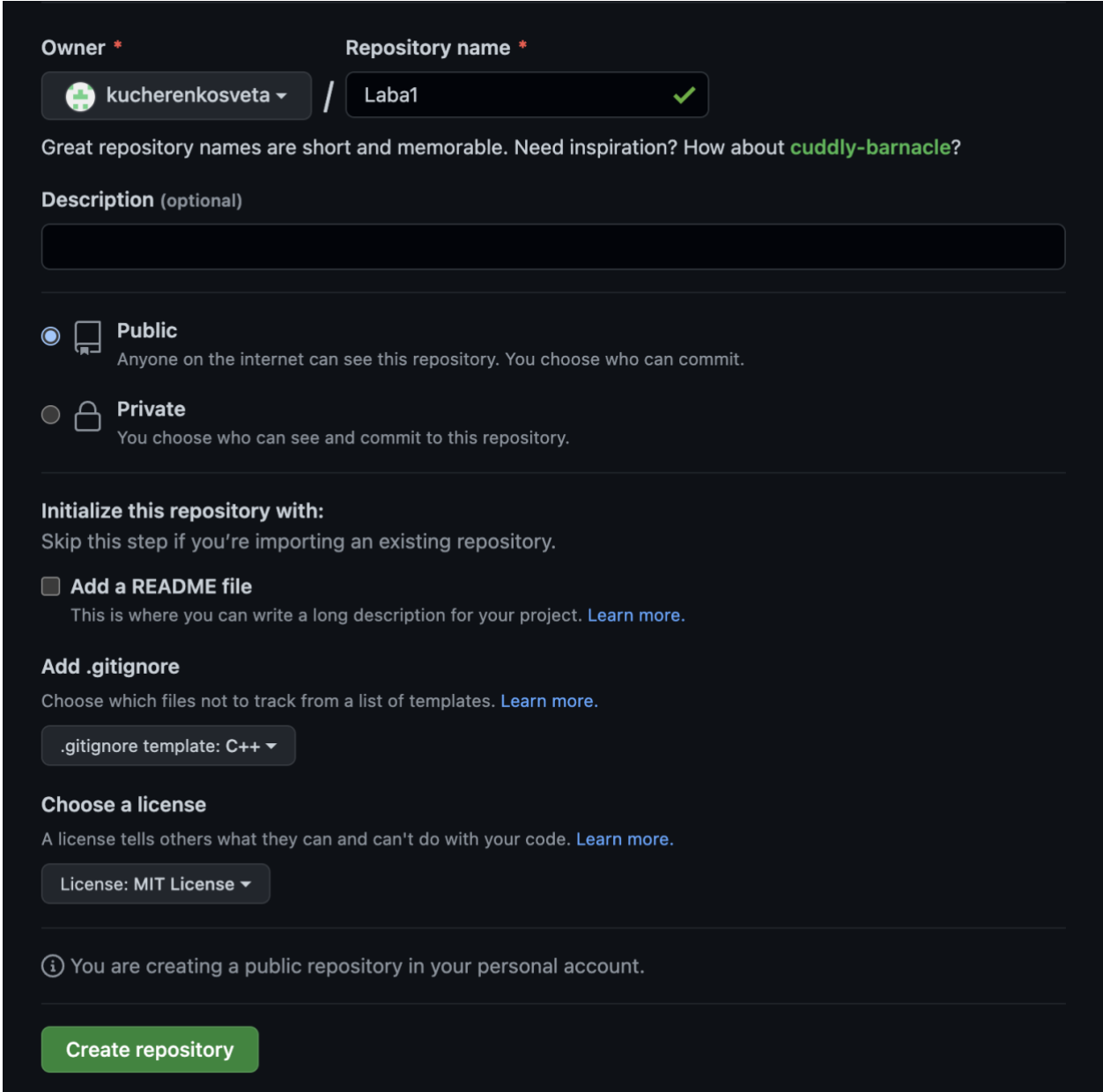


Рисунок 1.8 – Отправка изменений на сервер в GitHub Desktop

Практическая часть

(https://github.com/kucherenkosveta/LR_1.git)

Был изучен теоретический материал, после чего был создан общедоступный репозиторий на GitHub, в котором использована лицензия MIT и выбранный язык программирования (в моем случае C++).



The screenshot shows the GitHub 'Create repository' form. At the top, the 'Owner' is set to 'kucherenkosveta' and the 'Repository name' is 'Laba1', which is marked as valid with a green checkmark. Below this, a message suggests that great repository names are short and memorable, with an example 'cuddly-barnacle'. The 'Description' field is empty. Under the 'Visibility' section, 'Public' is selected, indicating that anyone on the internet can see the repository. The 'Private' option is also visible. The 'Initialize this repository with' section includes a checkbox for 'Add a README file' and a dropdown for '.gitignore template' set to 'C++'. The 'Choose a license' section has a dropdown set to 'MIT License'. A note at the bottom states, 'You are creating a public repository in your personal account.' A green 'Create repository' button is at the bottom left.

Owner * Repository name *

kucherenkosveta / Laba1 ✓

Great repository names are short and memorable. Need inspiration? How about **cuddly-barnacle**?

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: C++ ▾

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

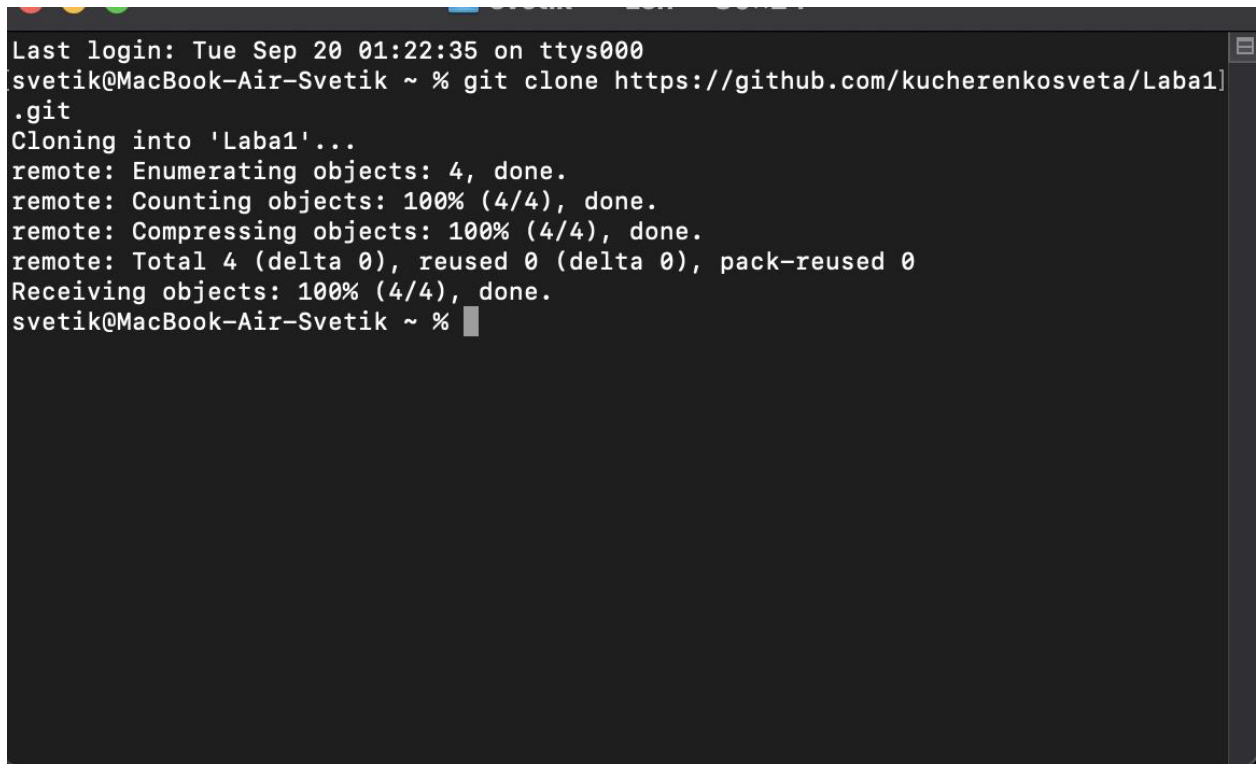
License: MIT License ▾

i You are creating a public repository in your personal account.

Create repository

Рисунок 2.1 – Создание репозитория

Далее было выполнено клонирование созданного репозитория на рабочий компьютер

A terminal window with a dark background. The text shows the execution of a git clone command to clone a repository from GitHub. The output indicates that 4 objects were enumerated, counted, and compressed, and that the cloning process was successful.

```
Last login: Tue Sep 20 01:22:35 on ttys000
svetik@MacBook-Air-Svetik ~ % git clone https://github.com/kucherenkosveta/Laba1
.git
Cloning into 'Laba1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
svetik@MacBook-Air-Svetik ~ %
```

Рисунок 2.2 – Клонирование репозитория

Далее в файл README.md была добавлена информация о группе и ФИО студента, выполняющего лабораторную работу.

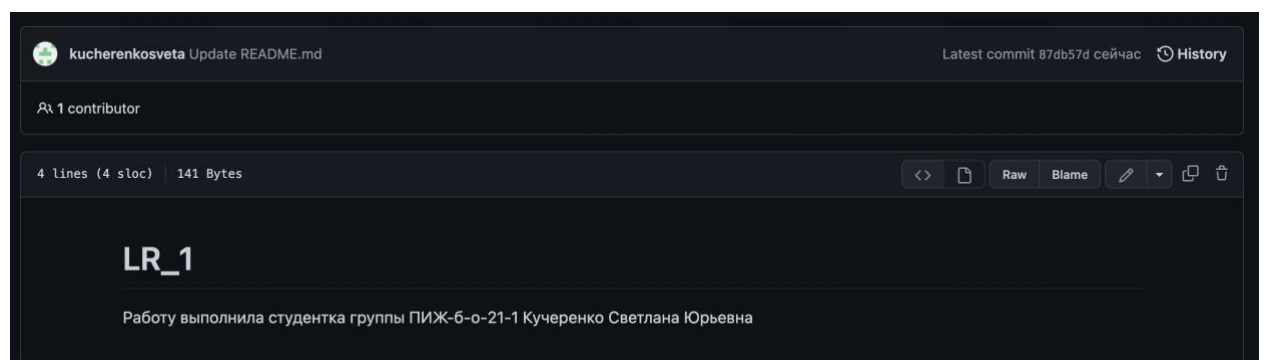


Рисунок 2.3 – Изменения в файле README.md

Была написана небольшая программа на языке C++, а также зафиксированы изменения при написании программы в локальном репозитории. Было сделано 7 коммитов.

```

 2 files changed, 1 insertion(+)
[svetik@MacBook-Air-Svetik LR_1 % git add .
[svetik@MacBook-Air-Svetik LR_1 % git commit -m "6save"
[main 2e54df2] 6save
 2 files changed, 1 insertion(+), 1 deletion(-)
[svetik@MacBook-Air-Svetik LR_1 % git add .
[svetik@MacBook-Air-Svetik LR_1 % git commit -m "7save"
[main e1bbd39] 7save
 2 files changed, 1 insertion(+), 1 deletion(-)
[svetik@MacBook-Air-Svetik LR_1 % git log
commit e1bbd3906e228c209a1fa8b3ed6f8a7ca36b143f (HEAD -> main)
Author: kucherenkosveta <kucherenko55sss55@gmail.com>
Date: Sat Sep 24 01:16:11 2022 +0300

    7save

commit 2e54df27ea830bad13269835d3f022c87a25e8c8
Author: kucherenkosveta <kucherenko55sss55@gmail.com>
Date: Sat Sep 24 01:15:41 2022 +0300

    6save

commit e0ce786811a9b6a3f8ec28143bc1a6def0d07173
Author: kucherenkosveta <kucherenko55sss55@gmail.com>
Date: Sat Sep 24 01:14:55 2022 +0300

    5save

commit aebdf230a37c819d240914e2836d9d413f409b6a
Author: kucherenkosveta <kucherenko55sss55@gmail.com>
Date: Sat Sep 24 01:10:13 2022 +0300

    4save

commit 14156150e88595f3283a6b4abbd44056299f72db
Author: kucherenkosveta <kucherenko55sss55@gmail.com>
Date: Sat Sep 24 01:08:59 2022 +0300

    3save

commit 670260ca6613b75a65fc9611f78cd7ba5a7d1dc4
Author: kucherenkosveta <kucherenko55sss55@gmail.com>
Date: Sat Sep 24 01:07:07 2022 +0300

    2save

commit 479e2a186c13492fa4d27ab2e11dbba4b301212a
Author: kucherenkosveta <kucherenko55sss55@gmail.com>
Date: Sat Sep 24 01:05:37 2022 +0300

    1save

commit 44f407d55b18d17a68f87aedd699129d08f19968 (origin/main, origin/HEAD)
Author: kucherenkosveta <112749526+kucherenkosveta@users.noreply.github.com>
Date: Sat Sep 24 00:47:22 2022 +0300

    Initial commit
[svetik@MacBook-Air-Svetik LR_1 % git status
On branch main
Your branch is ahead of 'origin/main' by 7 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

```

Рисунок 2.4 – Зафиксированные изменения

Далее был добавлен файл README.txt и зафиксированы сделанные изменения


 README.txt Create README.txt

Рисунок 2.5 – Добавление файла README

Отправлены изменения из локального репозитория в удаленный репозиторий GitHub.

```
[svetik@MacBook-Air-Svetik LR_1 % git push
Username for 'https://github.com': kucherenkosveta
[Password for 'https://kucherenkosveta@github.com':
Enumerating objects: 80, done.
Counting objects: 100% (80/80), done.
Delta compression using up to 8 threads
Compressing objects: 100% (63/63), done.
Writing objects: 100% (79/79), 29.04 KiB | 5.81 MiB/s, done.
Total 79 (delta 23), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (23/23), done.
To https://github.com/kucherenkosveta/LR_1.git
 44f407d..e1bbd39  main -> main
svetik@MacBook-Air-Svetik LR_1 %
```

Рисунок 2.6 - распространение изменений в исходном репозитории на GitHub

Проконтролированы изменения, произошедшие в репозитории GitHub.

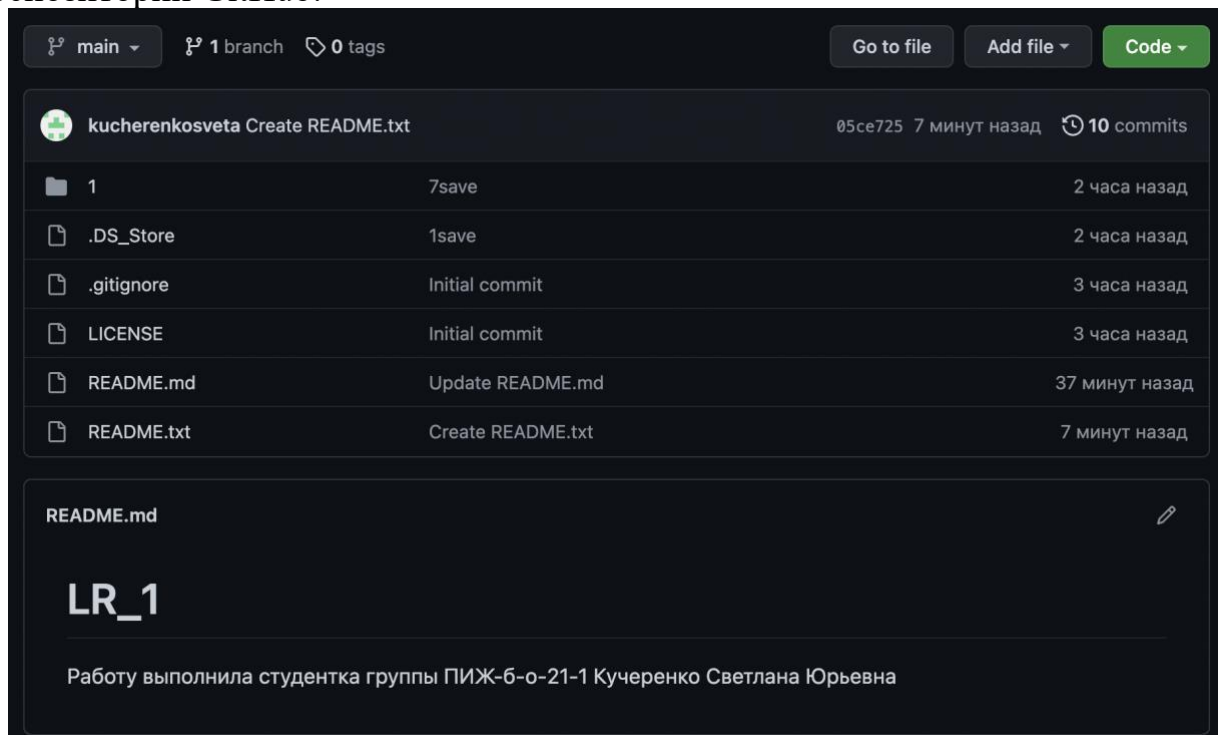


Рисунок 2.7 – Репозиторий на GitHub после изменений

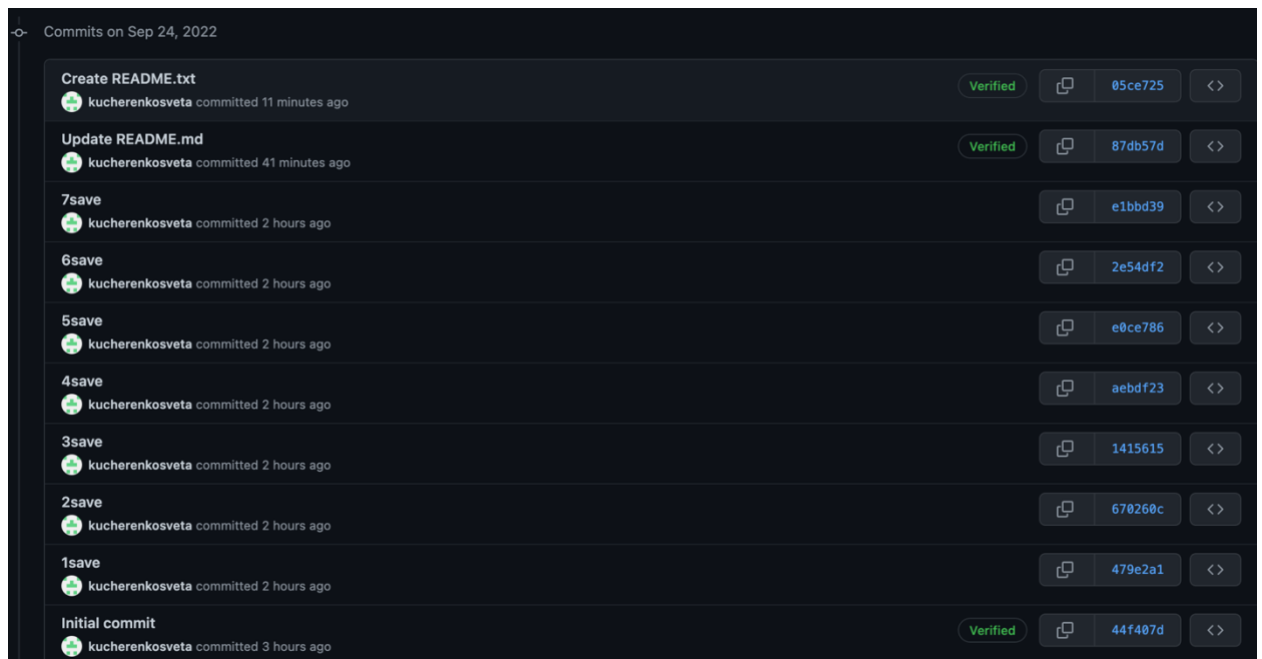


Рисунок 2.8 – Сделанные коммиты

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