

**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

Образцова М. Д. « 9 » 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/obrMaria/OPI_LR_1.git)

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 репозиториев.

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`

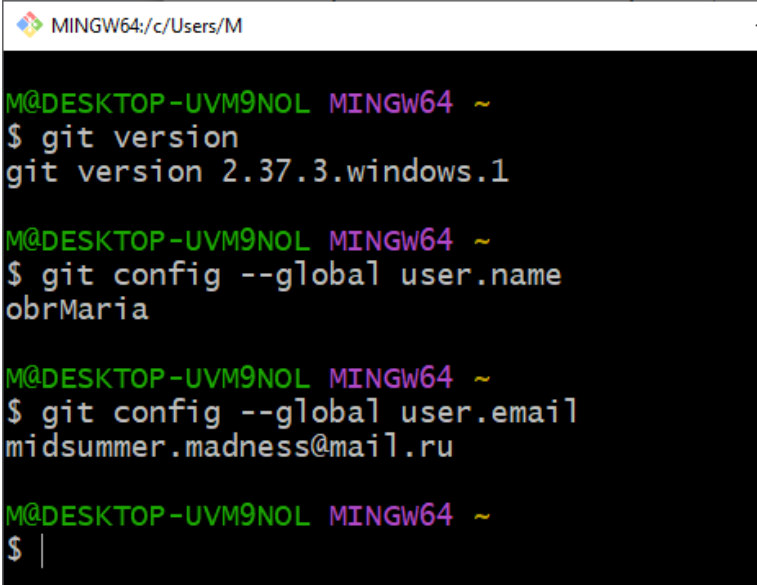
Удаленный репозиторий- это репозиторий, который хранится в облаке, на сторонних сервисах, специально созданных под работу с проектами `git`.

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:



```
MINGW64: c:/Users/M
M@DESKTOP-UVM9NOL MINGW64 ~
$ git version
git version 2.37.3.windows.1

M@DESKTOP-UVM9NOL MINGW64 ~
$ git config --global user.name
obrMaria

M@DESKTOP-UVM9NOL MINGW64 ~
$ git config --global user.email
midsummer.madness@mail.ru

M@DESKTOP-UVM9NOL MINGW64 ~
$ |
```

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.

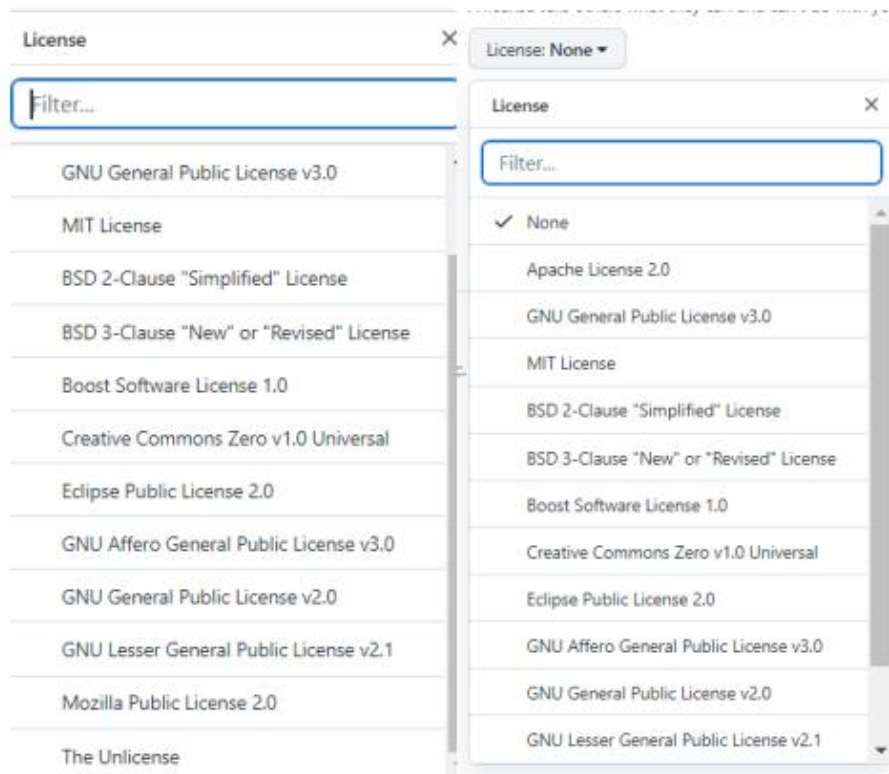


Рисунок 1.2– Лицензии GitHub

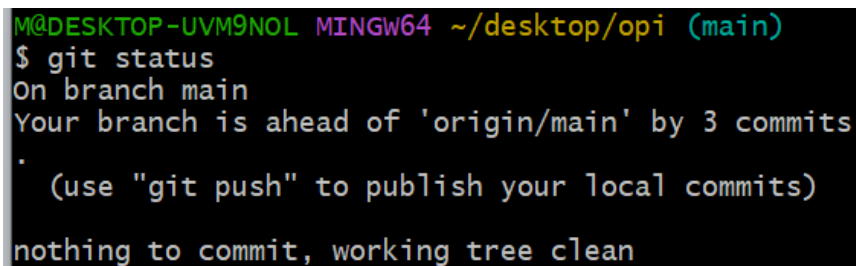
13. Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий?

```
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 запушить.

`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:

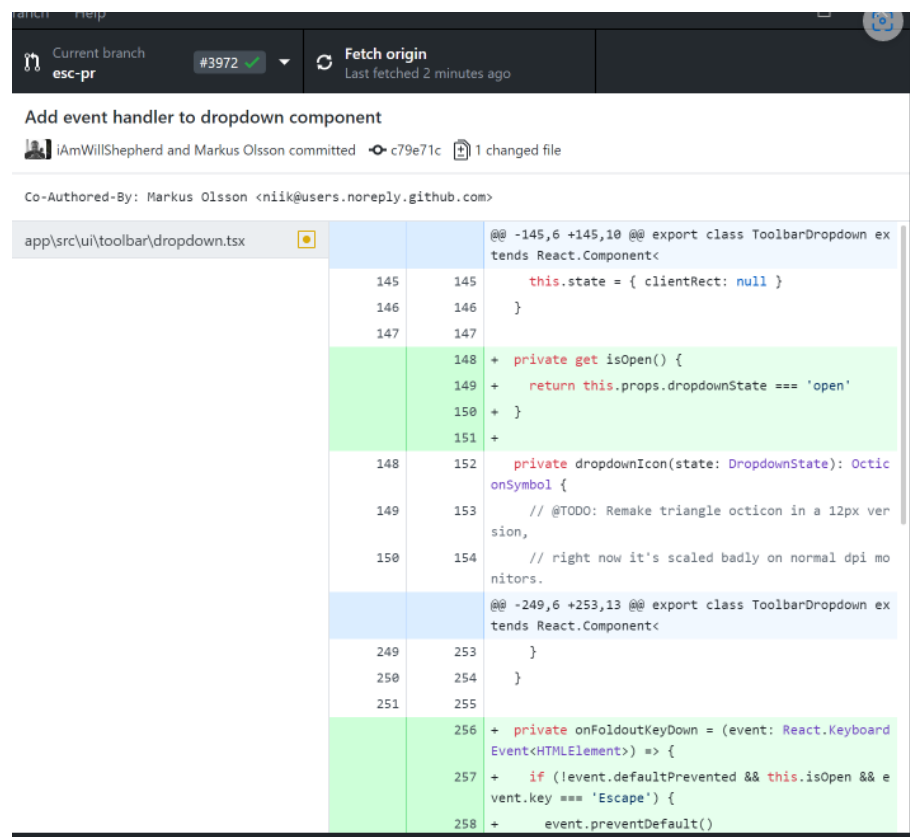


Рисунок 1.4 – История изменений в GitHub Desktop

Можно клонировать/ создать / добавить репозиторий.

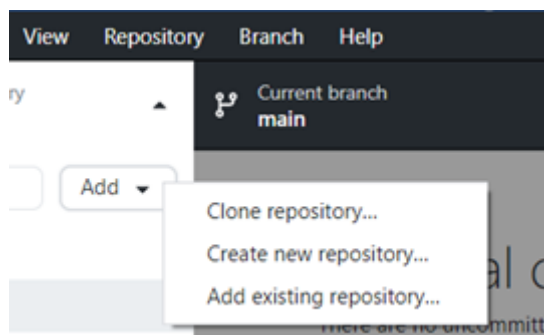


Рисунок 1.5 – Работа с репозиториями в GitHub Desktop

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

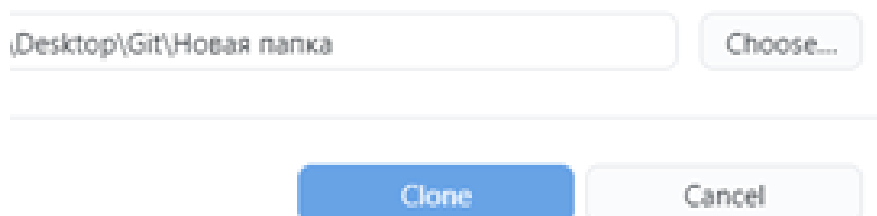


Рисунок 1.6 – Клонирование репозитория в GitHub Desktop

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

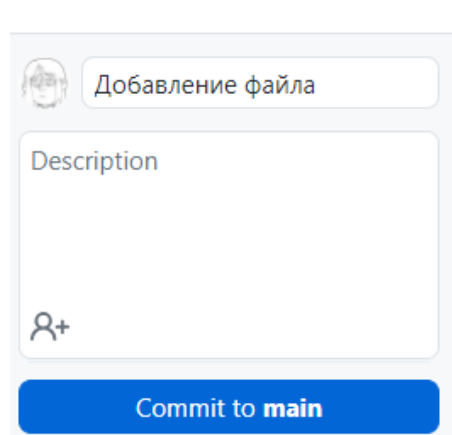


Рисунок 1.7 –коммит в GitHub Desktop

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

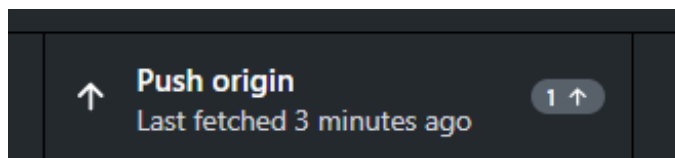


Рисунок 1.8 – Отправка изменений на сервер в GitHub Desktop

Практическая часть

(https://github.com/obrMaria/OPI_LR_1.git)

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

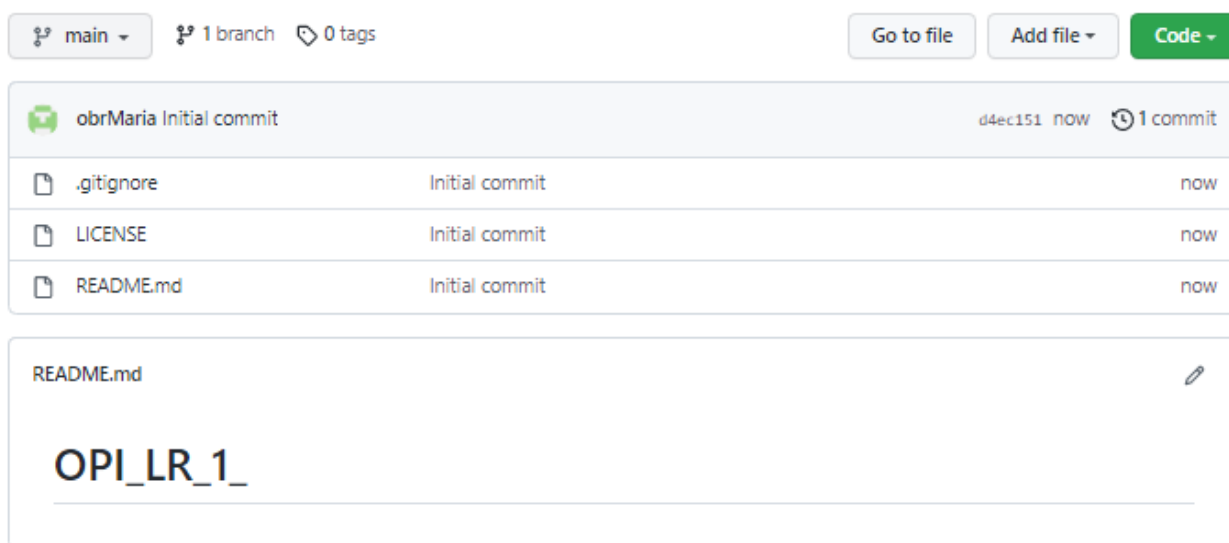


Рисунок 2.1 – Создание репозитория OPI_LR_1

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

```
M@DESKTOP-UVM9NOL MINGW64 ~  
$ pwd  
/c/Users/M  
  
M@DESKTOP-UVM9NOL MINGW64 ~  
$ cd desktop  
  
M@DESKTOP-UVM9NOL MINGW64 ~/desktop  
$ git clone https://github.com/obrMaria/OPI_LR_1.git  
Cloning into 'OPI_LR_1'...  
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.
```

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. Поэтому следующим шагом было дополнение файла .gitignore парой строк.

```
34 #дополнение .gitignore
35 ! *.png # НЕ исключать файлы с таким расширением
36 *.xlsx # исключать файлы с таким расширением
```

Рисунок 2.3 – Изменения внесенные в файл . gitignore

```
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 2.4 – Проверка статуса репозитория

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

OPI_LR_1 / README.md in main

<> Edit file Preview

```
1 # OPI_LR_1
2
3 работу выполнила студентка
4 группы ПИЖ-6-о-21-1
5 Образцова Мария Дмитриевна
6
```

Рисунок 2.5 – Изменения в файле README.md

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».

```

1@DESKTOP-A8I4D4M MINGW64 ~/OPI_LR_1 (main)
$ git log
commit 8281d977be0790208266827c219640cad5834fd (HEAD -> main, origin/main, origin/HEAD)
Merge: e0c935b b6c6124
Author: obrMaria <midsummer.madness@mail.ru>
Date: Sun Sep 11 13:51:07 2022 +0300

    Merge https://github.com/obrMaria/OPI_LR_1

commit e0c935b170726e4572ef52fef2d4d96893c9df7c
Author: obrMaria <midsummer.madness@mail.ru>
Date: Sun Sep 11 13:40:10 2022 +0300

    readme file

commit b6c61245aaea229a50b2f084600ee8d51b4d546c
Author: obrMaria <112749225+obrMaria@users.noreply.github.com>
Date: Sun Sep 11 13:14:32 2022 +0300

    commit with gitignore

commit 85a6a02e87e1ca9ea35043470ed4fcdd76607567
Author: obrMaria <112749225+obrMaria@users.noreply.github.com>
Date: Sun Sep 11 13:13:17 2022 +0300
:...skipping...
commit 8281d977be0790208266827c219640cad5834fd (HEAD -> main, origin/main, origin/HEAD)
Merge: e0c935b b6c6124
Author: obrMaria <midsummer.madness@mail.ru>
Date: Sun Sep 11 13:51:07 2022 +0300

    Merge https://github.com/obrMaria/OPI_LR_1

commit e0c935b170726e4572ef52fef2d4d96893c9df7c
Author: obrMaria <midsummer.madness@mail.ru>
Date: Sun Sep 11 13:40:10 2022 +0300

```

Рисунок 2.6 – Зафиксированные изменения

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

```

1@DESKTOP-A8I4D4M MINGW64 ~
$ echo "new README file" > README.txt

```

Рисунок 2.7 – Добавление файла README.txt

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

```

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_1 (main)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 26.99 KiB | 1.80 MiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/obrMaria/OPI_LR_1.git
9f4bd18..72ba758  main -> main

```

Рисунок 2.8 – распространение изменений в исходном репозитории на GitHub,

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

The screenshot shows a GitHub repository page for user 'obrMaria'. At the top, it indicates 'main' branch, '1 branch', and '0 tags'. There are buttons for 'Go to file', 'Add file', and 'Code'. Below this, a merge commit is shown: 'obrMaria Merge https://github.com/obrMaria/OPI_LR_1' with commit hash '8281d97', '9 hours ago', and '17 commits'. A table lists the files in the repository:

File	Commit	Time
folder_1	4 commit	2 days ago
folder_2/ConsoleApplication1	7 commit	2 days ago
otchet.doc	исправление отчета 1.1	2 days ago
.gitignore	Update .gitignore	10 hours ago
LICENSE	Initial commit	2 days ago
README.md	Update README.md	2 days ago
README.txt	readme file	10 hours ago

Below the table, the 'README.md' file is selected, showing its content:

```

OPI_LR_1

работу выполнила студентка группы ПИЖ-6-о-21-1 Образцова Мария Дмитриевна

```

Рисунок 2.9 – Страница на GitHub после внесенных изменений

main		
Commits on Sep 11, 2022		
Merge https://github.com/obrMaria/OPI_LR_1 obrMaria committed 9 hours ago	8281d97	<>
readme file obrMaria committed 10 hours ago	e8c935b	<>
commit with gitignore obrMaria committed 10 hours ago	Verified b6c6124	<>
Update .gitignore obrMaria committed 10 hours ago	Verified 85a6a82	<>
Update .gitignore obrMaria committed 10 hours ago	Verified 0a79e91	<>
Commits on Sep 10, 2022		
Merge branch 'main' of https://github.com/obrMaria/OPI_LR_1 obrMaria committed 2 days ago	5c07fae	<>
исправление отчета 1.1 obrMaria committed 2 days ago	3dd16f7	<>
коммит:добавление отчета в папку obrMaria committed 2 days ago	8cddead	<>
7 commit obrMaria committed 2 days ago	2e18353	<>
6 commit obrMaria committed 2 days ago	c8328c9	<>
5 commit obrMaria committed 2 days ago	b4c1098	<>

Рисунок 2.10 – Все коммиты, просмотренные в GitHub

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