

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Исследование основных возможностей Git и GitHub»

ОТЧЕТ
по лабораторной работе №1
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна

2 курс, группа ПИЖ-б-о-21-1,

09.03.04 «Программная инженерия»,

направленность (профиль) «Разработка

и сопровождение программного

обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2022 г.

Лабораторная работа 1.1 Исследование основных возможностей Git и GitHub.

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

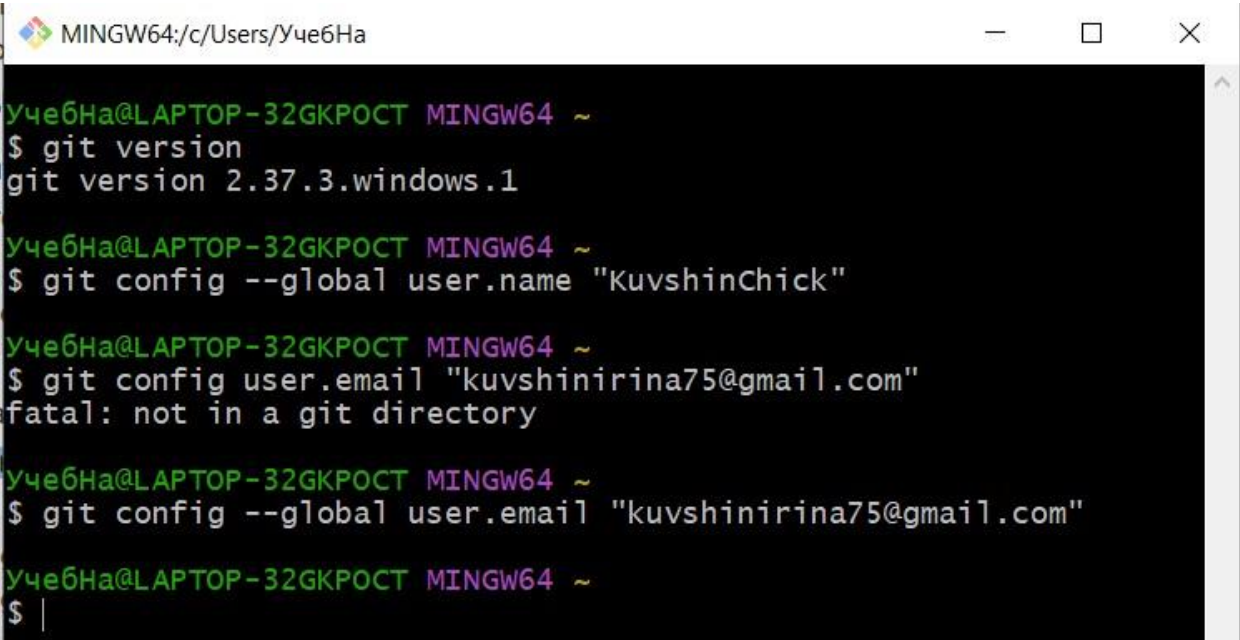
Ссылка на репозиторий: <https://github.com/KuvshinChick/TheFirstLab.git>

Ход работы:

1. Установка и настройка Git

При первоначальной настройке задается имя пользователя и адрес электронной почты, которыми будет идентифицироваться авторство коммитов в репозитории Git.

```
git config --global user.name "Your Name"
git config --global user.email "your_email@whatever.com"
```

A screenshot of a Windows command prompt window titled "MINGW64: c:/Users/УчебНа". The window shows the following commands and output:

```
УчебНа@LAPTOP-32GKPOCT MINGW64 ~
$ git version
git version 2.37.3.windows.1

УчебНа@LAPTOP-32GKPOCT MINGW64 ~
$ git config --global user.name "KuvshinChick"

УчебНа@LAPTOP-32GKPOCT MINGW64 ~
$ git config user.email "kuvshinirina75@gmail.com"
fatal: not in a git directory

УчебНа@LAPTOP-32GKPOCT MINGW64 ~
$ git config --global user.email "kuvshinirina75@gmail.com"

УчебНа@LAPTOP-32GKPOCT 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).)

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner * Repository name *

 KuvshinChick / TheFirstLab 

Great repository names are short and memorable. Need inspiration? How about [sturdy-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

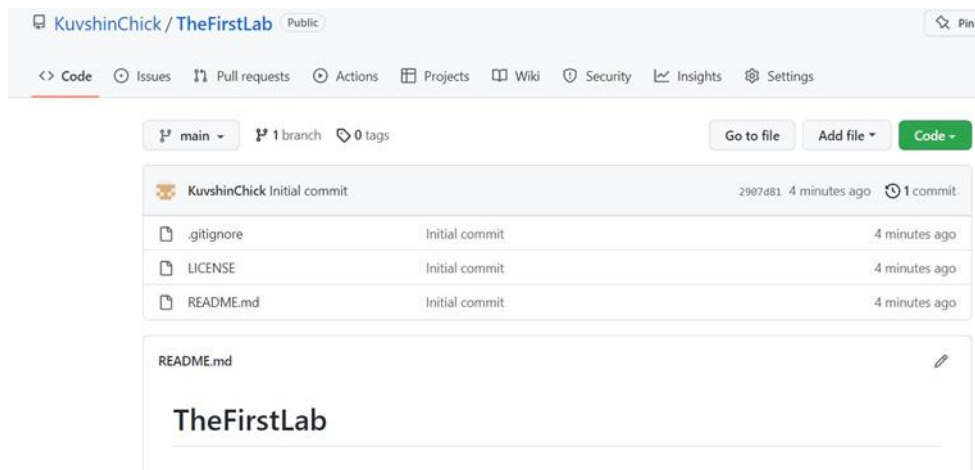
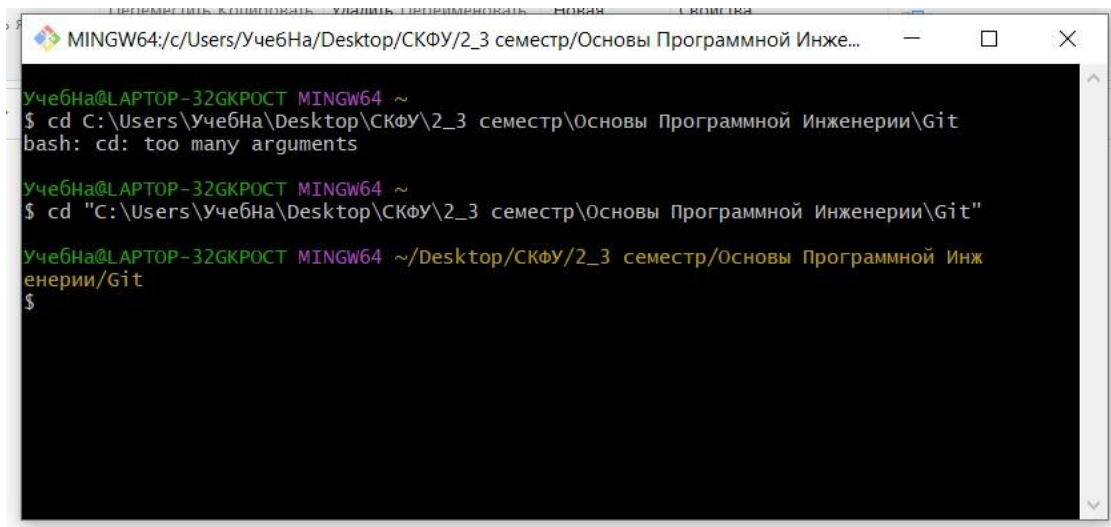


Рисунок 1.3 – Создание нового репозитория

3. Клонирование созданного репозитория на рабочий компьютер.

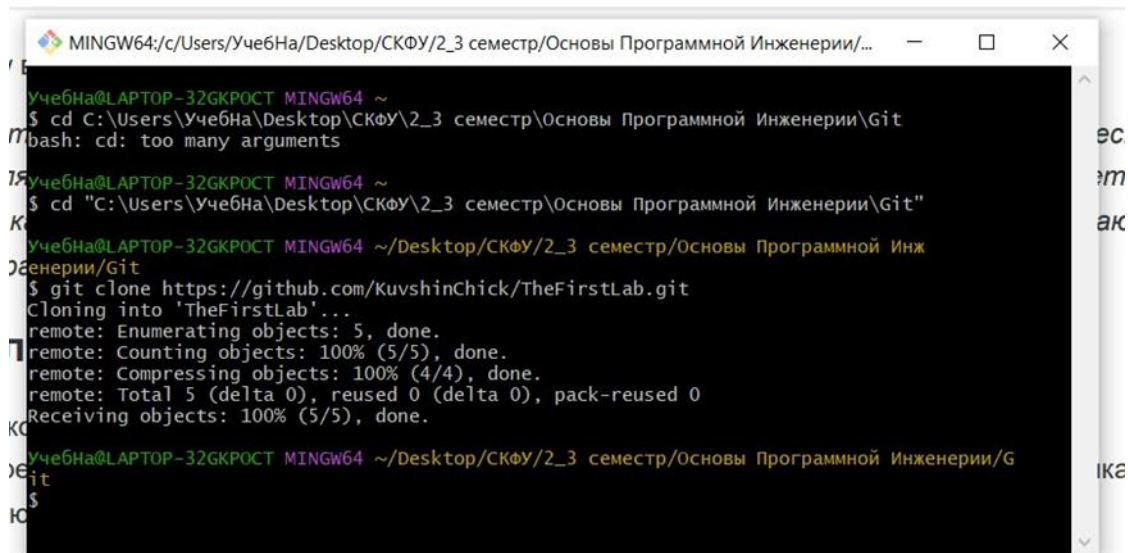


```
MINGW64/c:/Users/Учебна/Desktop/СКФУ/2_3 семестр/Основы Программной Инже...
Учебна@LAPTOP-32GKPOCT MINGW64 ~
$ cd C:\Users\Учебна\Desktop\СКФУ\2_3 семестр\Основы Программной Инженерии\Git
bash: cd: too many arguments

Учебна@LAPTOP-32GKPOCT MINGW64 ~
$ cd "C:\Users\Учебна\Desktop\СКФУ\2_3 семестр\Основы Программной Инженерии\Git"

Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3 семестр/Основы Программной Инж
енерии/Git
$
```

Рисунок 1.4 – Переход в каталог



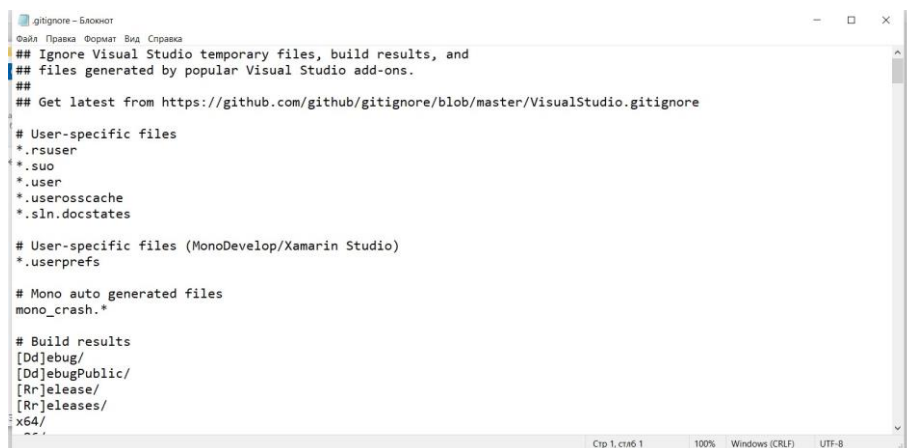
```
MINGW64/c:/Users/Учебна/Desktop/СКФУ/2_3 семестр/Основы Программной Инженерии/...
Учебна@LAPTOP-32GKPOCT MINGW64 ~
$ cd C:\Users\Учебна\Desktop\СКФУ\2_3 семестр\Основы Программной Инженерии\Git
bash: cd: too many arguments

Учебна@LAPTOP-32GKPOCT MINGW64 ~
$ cd "C:\Users\Учебна\Desktop\СКФУ\2_3 семестр\Основы Программной Инженерии\Git"

Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3 семестр/Основы Программной Инж
енерии/Git
$ git clone https://github.com/KuvshinChick/TheFirstLab.git
Cloning into 'TheFirstLab'...
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.

Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3 семестр/Основы Программной Инженерии/G
it
$
```

Рисунок 1.5 – Клонирование репозитория



```
.gitignore - Блокнот
Файл Правка Формат Вид Справка
## 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/
x64/
~\
```

Рисунок 1.6 – Файл «.gitignore»

README – Блокнот

Файл Правка Формат Вид Справка

TheFirstLab

Кувшин Ирина Анатольевна

ПИЖ-6-о-21-1 (1)

Рисунок 1.7 – Файл «ReadMe»

• Отправка изменения в удаленный репозиторий

Переход в локальную папку → внесение в индекс — временное хранилище — изменений, которые затем войдут в коммит (`git add .` - Вносит в индекс все изменения, включая новые файлы) → `git commit` (совершение коммита) → `git status` (состояние проекта) → `git push` (вносим изменения в удаленный репозиторий)

```
MINGW64/c:/Users/УчебHa/Desktop/Скøy/2.3 семестр/Основы Программной Инженерии/Git/TheFirstLab
УчебHa\ЛАРТОР-32GKPOCT MINGW64 ~
$ cd "c:/Users/УчебHa/Desktop/Скøy/2.3 семестр/Основы Программной Инженерии/Git/TheFirstLab"
УчебHa\ЛАРТОР-32GKPOCT MINGW64 ~/Desktop/Скøy/2.3 семестр/Основы Программной Инженерии/Git/TheFirstLab (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)
        modified:   README.md

УчебHa\ЛАРТОР-32GKPOCT MINGW64 ~/Desktop/Скøy/2.3 семестр/Основы Программной Инженерии/Git/TheFirstLab (main)
$ git commit -m "2 коммит RM"
[main 1473851] 2 коммит RM
1 file changed, 1 deletion(-)

УчебHa\ЛАРТОР-32GKPOCT MINGW64 ~/Desktop/Скøy/2.3 семестр/Основы Программной Инженерии/Git/TheFirstLab (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

УчебHa\ЛАРТОР-32GKPOCT MINGW64 ~/Desktop/Скøy/2.3 семестр/Основы Программной Инженерии/Git/TheFirstLab (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 297 bytes | 297.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/KuvshinChick/TheFirstLab.git
   ab06b0d..1473851  main -> main

УчебHa\ЛАРТОР-32GKPOCT MINGW64 ~/Desktop/Скøy/2.3 семестр/Основы Программной Инженерии/Git/TheFirstLab (main)
$
```

Рисунок 1.8 – Отправка изменения в удаленный реп

main 1 branch 0 tags

Go to file Add file Code

KuvshinChick RM changed 2.0 7438aae 18 seconds ago 9 commits

ConsoleApp1	Add some nums	4 minutes ago
.gitignore	Initial commit	2 hours ago
LICENSE	Initial commit	2 hours ago
README.md	RM changed 2.0	18 seconds ago

README.md

TheFirstLab

Кувшин Ирина Анатольевна ПИЖ-6-о-21-1 (1)

Рисунок 1.9 – Репозиторий с 9ю коммитами

```
MINGW64/c/Users/УчебНа/Desktop/Скøy/2_3 семестр/Основы Программной Инженерии/Git/TheFirstLab
УчебНа\ЛАТОР-32ГКРОСТ MINGW64 ~
$ cd "C:\Users\УчебНа\Desktop\Скøy\2_3 семестр\Основы Программной Инженерии\Git\TheFirstLab"
УчебНа\ЛАТОР-32ГКРОСТ MINGW64 ~/Desktop/Скøy/2_3 семестр/Основы Программной Инженерии/Git/TheFirstLab (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)
  "\320\233\320\240\1.docx"

nothing added to commit but untracked files present (use "git add" to track)
УчебНа\ЛАТОР-32ГКРОСТ MINGW64 ~/Desktop/Скøy/2_3 семестр/Основы Программной Инженерии/Git/TheFirstLab (main)
$ git add .
УчебНа\ЛАТОР-32ГКРОСТ MINGW64 ~/Desktop/Скøy/2_3 семестр/Основы Программной Инженерии/Git/TheFirstLab (main)
$ git commit -m "4otchet"
[main ef9eb6a] 4otchet
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 "\320\233\320\240\1.docx"
УчебНа\ЛАТОР-32ГКРОСТ MINGW64 ~/Desktop/Скøy/2_3 семестр/Основы Программной Инженерии/Git/TheFirstLab (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), 420.73 KiB | 28.05 MiB/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/KuvshinChick/TheFirstLab.git
   7d38aae..ef9eb6a  main -> main
УчебНа\ЛАТОР-32ГКРОСТ MINGW64 ~/Desktop/Скøy/2_3 семестр/Основы Программной Инженерии/Git/TheFirstLab (main)
$
```

Рисунок 1.10 – Добавление начального отчета в формате .doc (был удален)

Вопросы:

1. Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

2. В чем недостатки локальных и централизованных СКВ?

Локальная СКВ не предназначена для командной работы над проектом, т.к. представляет собой БД, хранящую записи обо всех изменениях в файлах.

Возможность потери данных.

Такие системы, как CVS, Subversion и Perforce, используют единственный сервер, содержащий все версии файлов, и некоторое количество клиентов, которые получают файлы из этого централизованного хранилища. ->

Единая точка отказа (— отсутствие доступа к данным при сбое работы сервера)

Возможность безвозвратной потери данных

3. К какой СКВ относится Git?

Для устранения единой точки отказа используются распределенные системы контроля версий. Они подразумевают, что клиент выкачивает себе весь репозиторий целиком вместо скачки конкретных интересующих клиента файлов. Если умрет любая

копия репозитория, то это не приведет к потере кодовой базы, поскольку она может быть восстановлена с компьютера любого разработчика. Каждая копия является полным бэкапом данных.

4. В чем концептуальное отличие Git от других СКВ?

Git стоит отдельно от других СКВ из-за подхода к работе с данными. Большинство других систем хранят информацию в виде списка изменений в файлах. Вместо этого, подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.

5. Как обеспечивается целостность хранимых данных в Git?

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом.

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

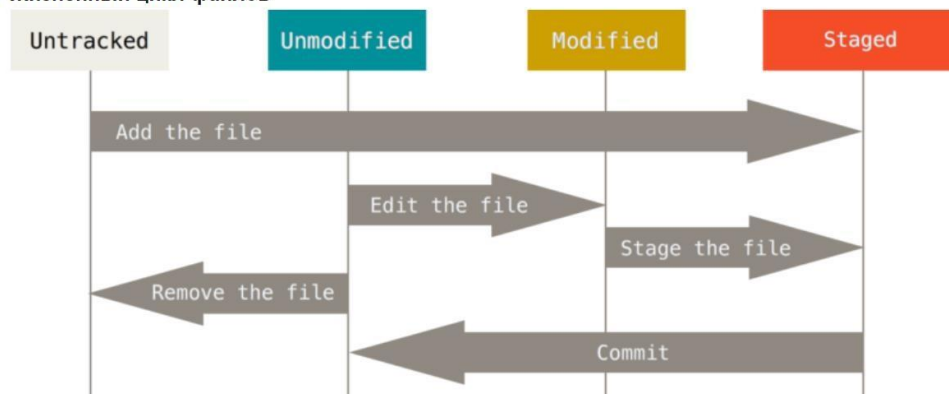
У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged).

Зафиксированный значит, что файл уже сохранён в вашей локальной базе.

К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы.

Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

Жизненный цикл файлов



Индекс — промежуточное место между вашим прошлым коммитом и следующим. Вы можете добавлять или удалять файлы из индекса. Когда вы делаете коммит в него попадают данные из индекса, а не из рабочей области.

Базовый подход в работе с Git выглядит так:

1. Вы изменяете файлы в вашей рабочей директории.
2. . Вы выборочно добавляете в индекс только те изменения, которые должны попасть в следующий коммит, добавляя тем самым снимки только этих изменений в область подготовленных файлов.
3. 3. Когда вы делаете коммит, используются файлы из индекса как есть, и этот снимок сохраняется в вашу Git-директорию.

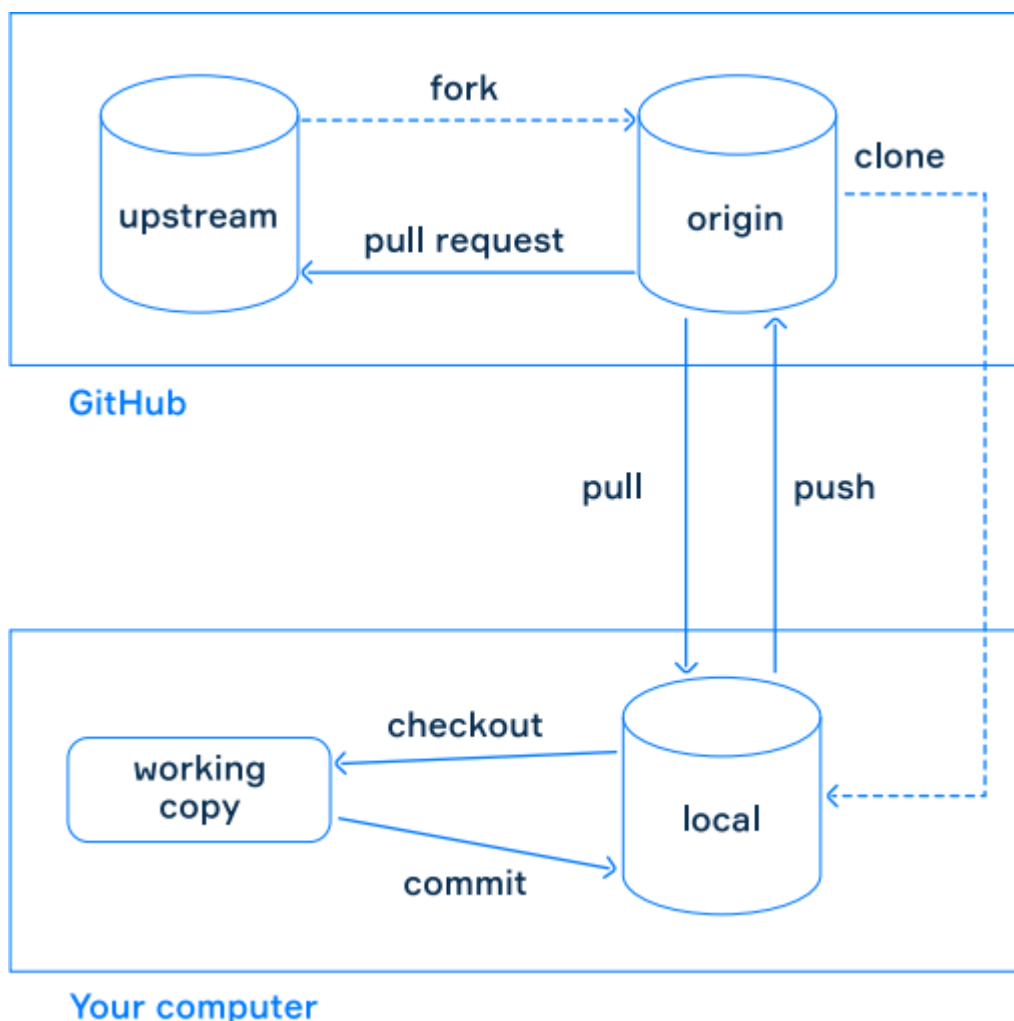
7. Что такое профиль пользователя в GitHub?

Аккаунт программиста, где он может хранить свои проекты и работать совместно с другими.

8. Какие бывают репозитории в GitHub?

Код, инструменты тестирования, базы знаний, шаблоны, справочники...

9. Укажите основные этапы модели работы с GitHub.



Стандартный подход к работе с проектом состоит в том, чтобы иметь локальную копию репозитория и фиксировать ваши изменения в этой копии, а не в удаленном репозитории, размещенном на GitHub. Этот локальный репозиторий имеет полную историю версий проекта, которая может быть полезна при разработке без подключения к интернету. После того, как вы что-то изменили в локальном, вы можете отправить свои изменения в удаленный репозиторий, чтобы сделать их видимыми для других разработчиков

Сначала рассмотрим область GitHub. В нем есть два хранилища:

upstream - это оригинальный репозиторий проекта, который вы скопировали,

origin - ваш fork (копия) на GitHub, к которому у вас есть полный доступ.

Чтобы перенести изменения с вашей копии в исходному репозиторий проекта, вам нужно сделать запрос на извлечение. Если вы хотите внести небольшие изменения в свою копию (fork), вы можете использовать веб-интерфейс GitHub. Однако такой подход не удобен при разработке программ, поскольку вам часто приходится запускать и отлаживать их локально. Стандартный способ - создать локальный клон удаленного репозитория и работать с ним локально, периодически внося изменения в удаленный репозиторий.

10. Как осуществляется первоначальная настройка Git после установки?

Устанавливается имя и почта пользователя, чтобы когда вы создавали commit, указывался автор, кто его создал.

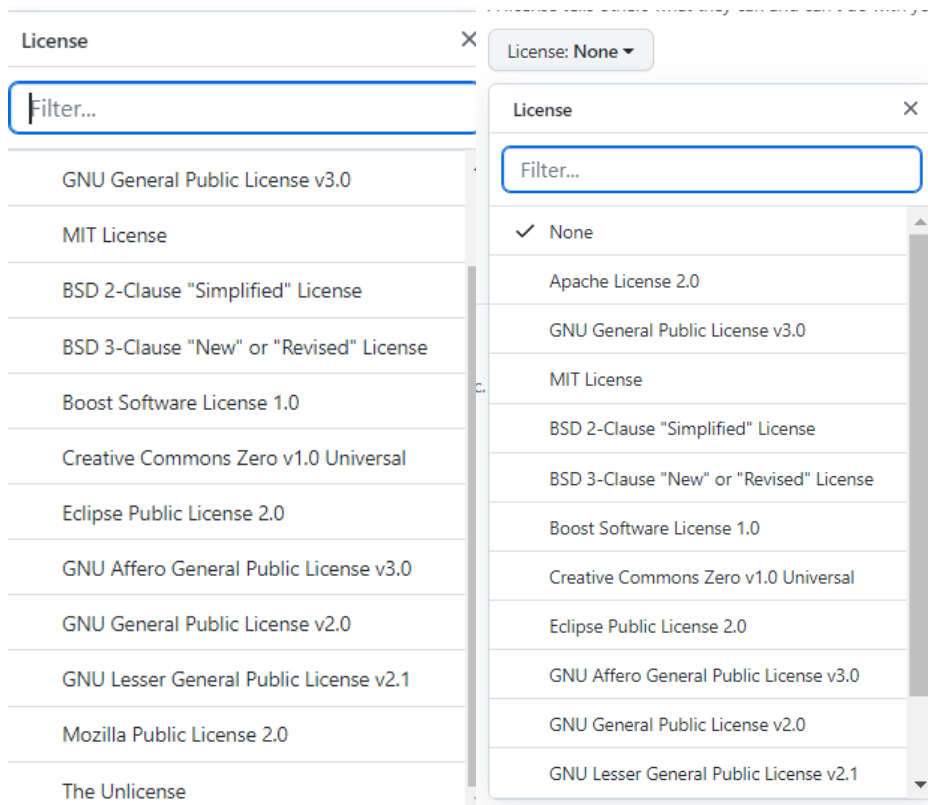
```
#Установим имя для вашего пользователя
#Вместо <ваше_имя> можно ввести, например, Grisha_Popov
#Кавычки оставляем
git config --global user.name "<ваше_имя>"

#Теперь установим email. Принцип тот же.
git config --global user.email "<адрес_почты@email.com>"
```

11. Опишите этапы создания репозитория в GitHub.

В профиле GitHub создать и настроить репозиторий, затем клонировать его на ПК, чтобы создать локальное хранилище проекта.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?



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

git clone <https://github.com/KuvshinChick/TheFirstLab.git>

Стандартный подход к работе с проектом состоит в том, чтобы иметь локальную копию репозитория и фиксировать ваши изменения в этой копии, а не в удаленном репозитории, размещенном на GitHub. Этот локальный репозиторий имеет полную историю версий проекта, которая может быть полезна при разработке без подключения к интернету. После того, как вы что-то изменили в локальном, вы можете отправить свои изменения в удаленный репозиторий, чтобы сделать их видимыми для других разработчиков.

14. Как проверить состояние локального репозитория Git?

git status

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 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.

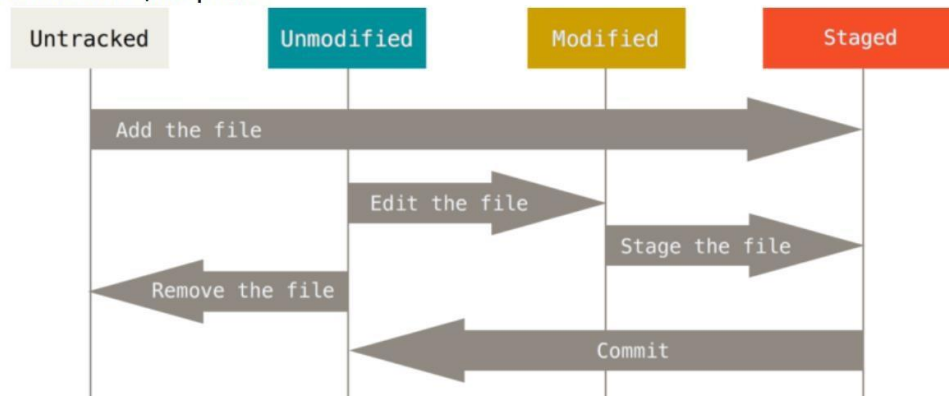
Если вы хотите распространить их в исходный репозиторий на GitHub, вам нужно использовать `push`. В первый раз вам также необходимо отправить свою локальную ветку, потому что она не существует в удаленном репозитории.

```
git push --set-upstream origin edit-readme
```

В следующий раз сделать

```
git push
```

Жизненный цикл файлов



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` . [на оба компьютера](#)

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

создать проект на GitHub (новый репозиторий) и скопировать его URL

перейти в Терминале в общую папку для проектов (папку самого проекта создавать не надо)

клонировать проект с GitHub — `git clone <URL>` (URL без скобок)

Чтобы локальные изменения (в данном примере — обновление файла и добавление картинки) синхронизировались с удалённым репозиторием, необходимо их сначала добавить, потом закоммитить, а потом запустить.



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

Bitbucket.

Bitbucket по умолчанию поддерживает Git

особенности:

- неограниченные частные репо;
- сравнение ветвей и история коммитов;
- клиент Bitbucket для Mac и Windows под названием SourceTree;
- приложение для Android под названием BitBeaker;
- Bitbucket для предприятий, называемый Stash;
- интеграция с такими инструментами, как Jira, Crucible, Bamboo, Jenkins,

HipChat;

- глубокая интеграция с Trello с помощью их возможности Bitbucket Cloud, которая предлагает бесшовную интеграцию ветвей, коммитов и pull-запросов в [доски Trello](#);

- разрешения на ветку – вместо того, чтобы предоставлять разработчикам доступ к каждой ветке в репо, Bitbucket позволяет ограничить доступ к одной ветке;

- простая интеграция с Bamboo и Confluence в дополнение к собственному Jira Software Cloud от Atlassian для дополнительной настройки и хостинга;

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).

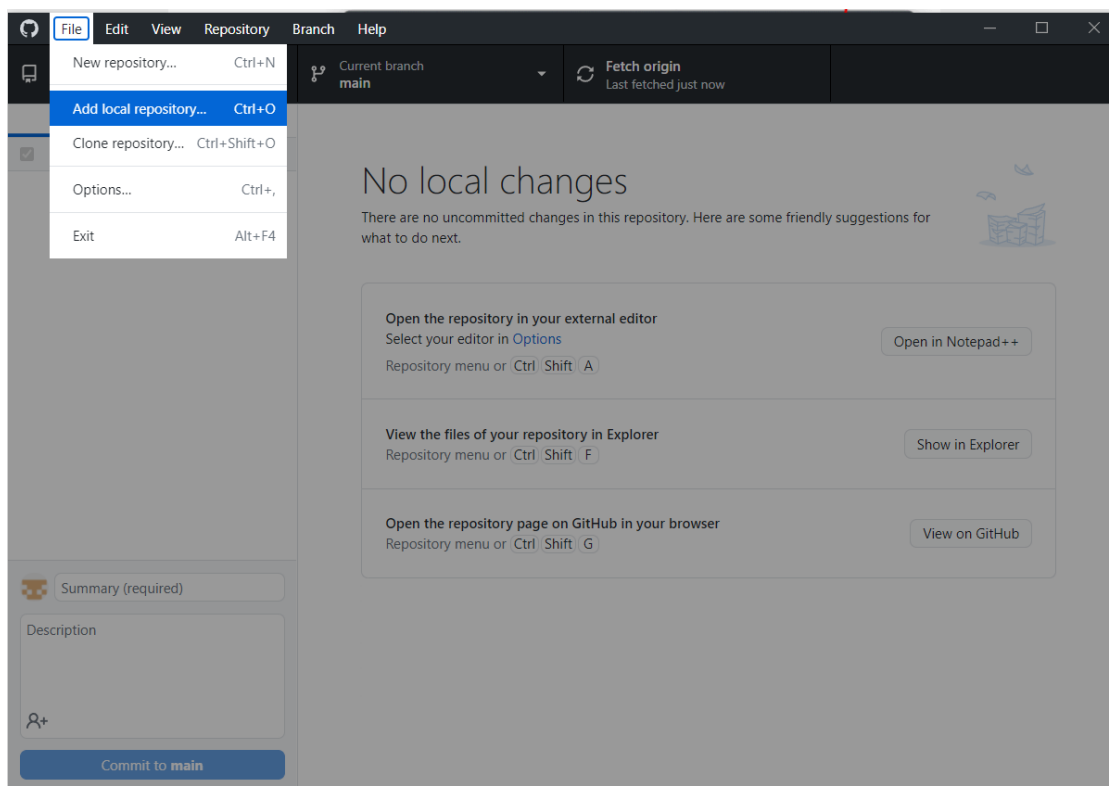


Рисунок 1.11 – Добавление локального репозитория

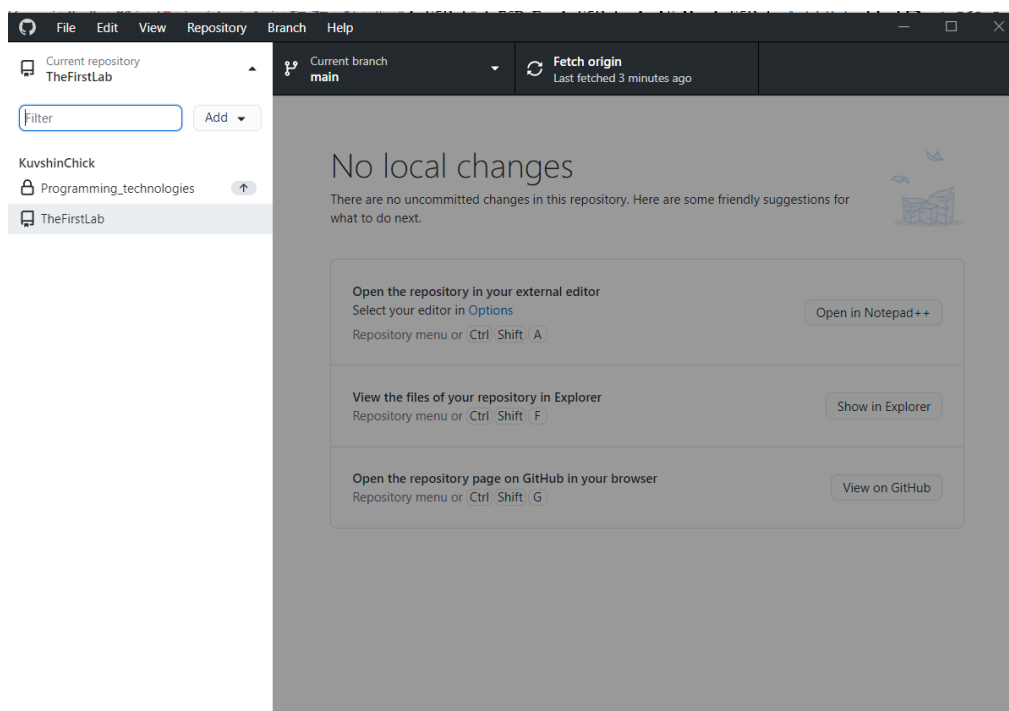


Рисунок 1.12 – Навигация среди репозиторияев

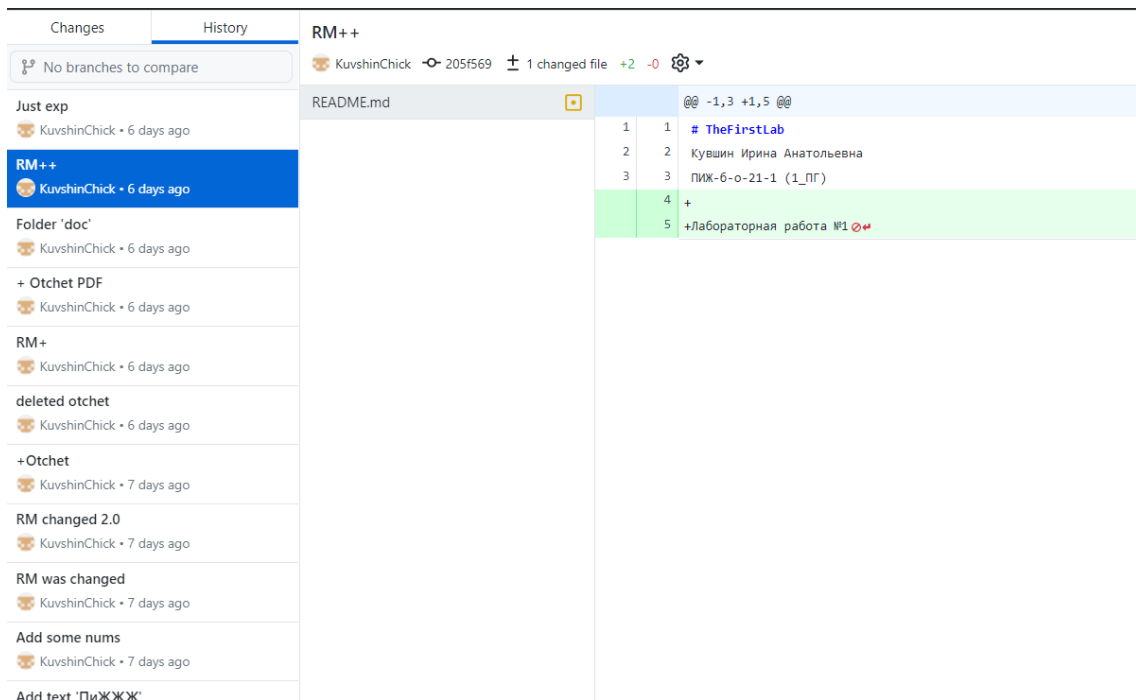


Рисунок 1.13 – История коммитов

Вывод: В ходе данной лабораторной работы были исследованы базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.