

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

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**  
**дисциплины «Основы программной инженерии»**

Выполнила:  
Панюкова Ксения Юрьевна  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка  
и сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р. А., доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

# Ход работы

## 1. Я изучила теоретический материал работы

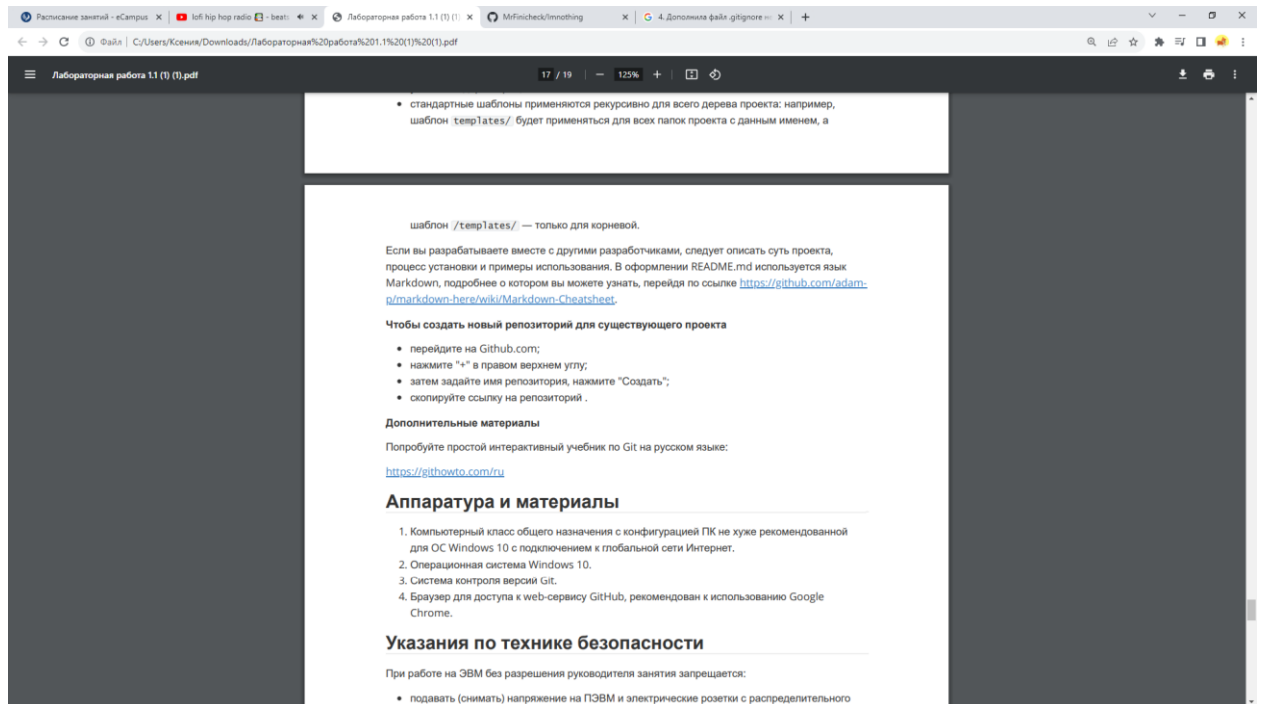


Рисунок 1.1 – Изучение материала для лабораторной работы

2. Создала общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный мною язык программирования

Лабораторная работа 1.1 (1) (1)

New repository

Новая вкладка

Create a new repository

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

Required fields are marked with an asterisk (\*).

Owner \*

Repository name \*

MrFinichck

/ Imnothing

Imnothing is available.

Great repository names are short and memorable. Need inspiration? How about **congenial-fiesta** ?

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:

☐ Add a README file

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

Add .gitignore

.gitignore template: Python

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

Choose a license

License: MIT License

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

You are creating a public repository in your personal account.

Create repository

Рисунок 2.1 – Настройка репозитория

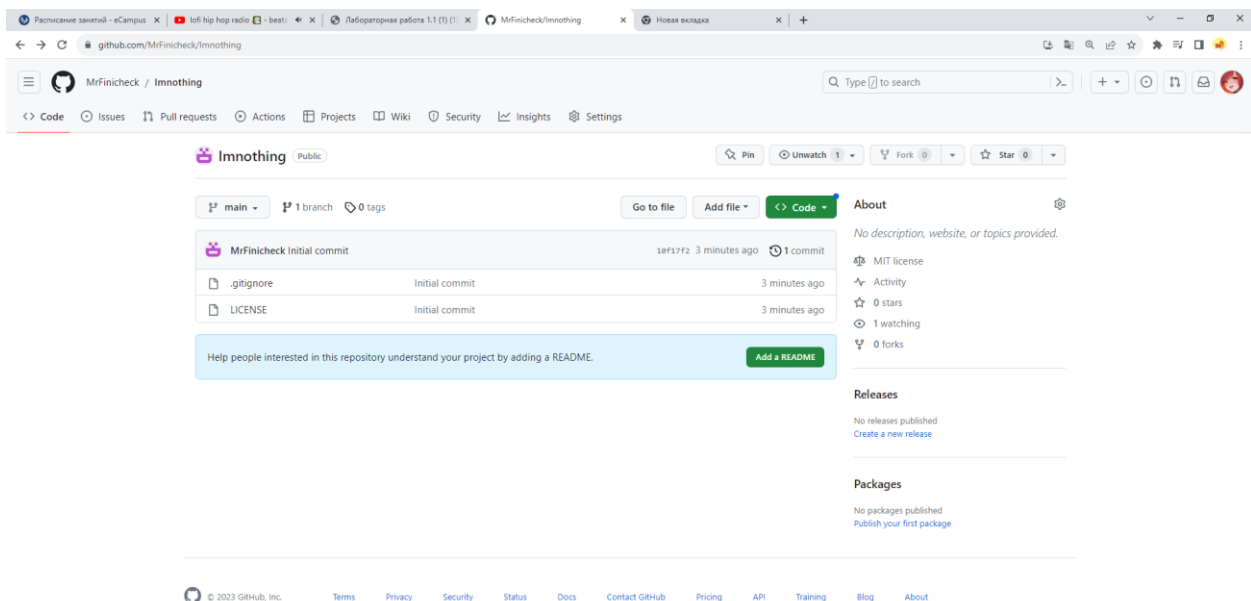


Рисунок 2.2 – Готовый репозиторий

3. Выполнила клонирование созданного репозитория на рабочий компьютер

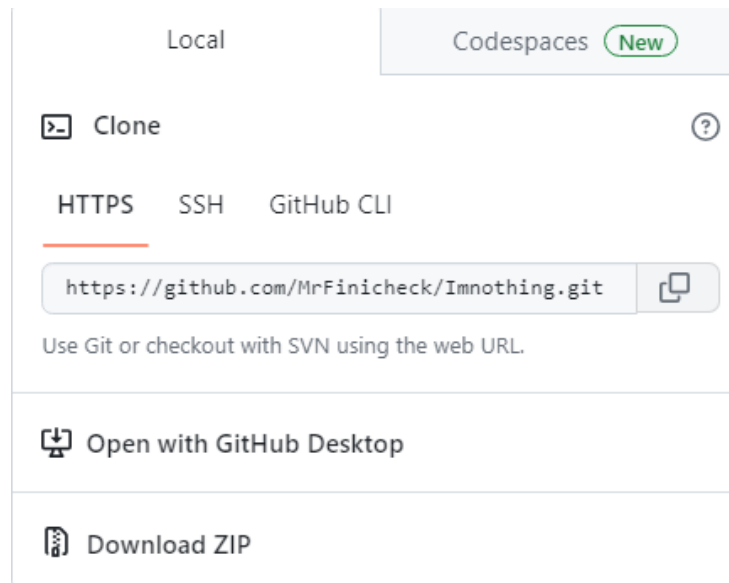


Рисунок 3.1 – Копирование ссылки репозитория

```
Microsoft Windows [Version 10.0.19044.1826]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Ксения>cd ..
C:\Users>cd ..
C:\>cd TestGit
C:\TestGit>mk dir NOTHING
"mk" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.
C:\TestGit>mkdir NOTHING
C:\TestGit>cd NOTHING
C:\TestGit\NOTHING>git clone https://github.com/MrFinicheck/Imnothing.git
Cloning into 'Imnothing'...
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.
C:\TestGit\NOTHING>
```

Рисунок 3.2 – Копирование репозитория на рабочий компьютер

4. Дополнила файл .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки

```

1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[co]
4  *.egg-info
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 -eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28
29 # PyInstaller
30 # Usually these files are written by a python script from a template
31 # before PyInstaller builds the exe, so as to inject date/other infos into it.
32 *.manifest
33 *.spec
34
35 # Installer logs
36 pip-log.txt
37 pip-delete-this-directory.txt
38
39 # Unit test / coverage reports
40 htmlcov/

```

Рисунок 4.1 – Файл с необходимыми правилами для языка Python

## 5. Добавила в файл README.md информацию о своей группе и моём ФИО

```

Администратор: Командная строка

nothing added to commit but untracked files present (use "git add" to track)
C:\TestGit\NOTHING\Imnothing>git add .
C:\TestGit\NOTHING\Imnothing>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.md

C:\TestGit\NOTHING\Imnothing>git commit -m "Add information about local repository in readme file"
[main 8eb6ce6] Add information about local repository in readme file
 1 file changed, 2 insertions(+)
 create mode 100644 README.md

C:\TestGit\NOTHING\Imnothing>git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 423 bytes | 423.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MrFinicheck/Imnothing.git
 10f17f2..8eb6ce6  main -> main
C:\TestGit\NOTHING\Imnothing>

```

Рисунок 5.1 – Добавляю в файл README.md информацию о своей группе и моём ФИО на GitHub через командную строку

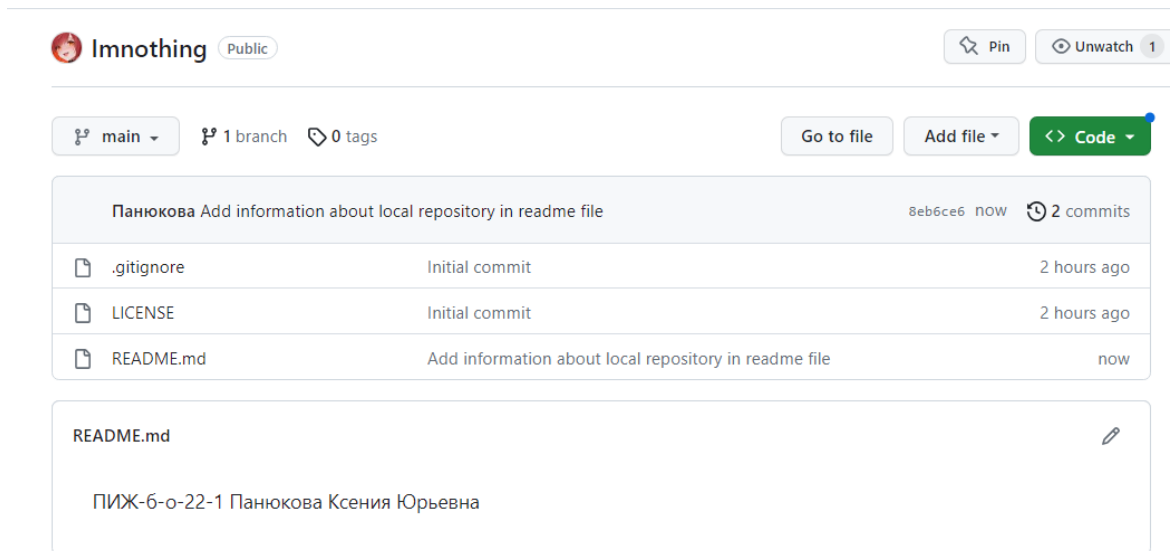


Рисунок 5.2 – Мой файл README.md на GitHub

6. Написала небольшую программу на выбранном мною языке программирования. Зафиксировала изменения при написании программы в локальном репозитории. Сделала не менее 7 коммитов

```
print("ДАВАЙ ЗНАКОМИТЬСЯ. НАПИШИ СВОЁ ИМЯ")
name = input()
time.sleep(2)
```

Рисунок 6.1 – Моя программа на Python

```
Файл Правка Формат Вид Справка
import time
print("ПРИВЕТ")
time.sleep(2)
print("ДАВАЙ ЗНАКОМИТЬСЯ. НАПИШИ СВОЁ ИМЯ")
name = input()
time.sleep(2)
print("Хех, здравствуй " + name + "!")
time.sleep(2)
```

Рисунок 6.2 – Изменяю программу

```

C:\TestGit\NOTHING\Imnothing>git commit -m"
C:\TestGit\NOTHING\Imnothing>
C:\TestGit\NOTHING\Imnothing>git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
    add

C:\TestGit\NOTHING\Imnothing>git add .

C:\TestGit\NOTHING\Imnothing>git commit -m"The robot is glad that he made friends with you"
[main ab2810e] The robot is glad that he made friends with you
 1 file changed, 6 insertions(+)

C:\TestGit\NOTHING\Imnothing>git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 4 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (12/12), 1.63 KiB | 1.63 MiB/s, done.
Total 12 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), completed with 1 local object.
To https://github.com/MrFinicheck/Imnothing.git
   3ac8411..ab2810e  main -> main

C:\TestGit\NOTHING\Imnothing>

```

Рисунок 6.3 – Делаю 7 коммитов

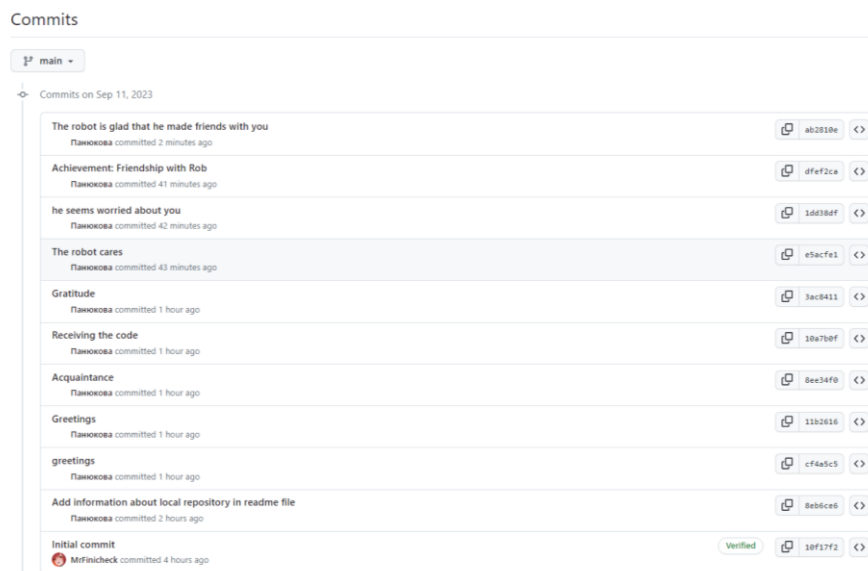


Рисунок 6.4 – Изменения, отображённые на GitHub

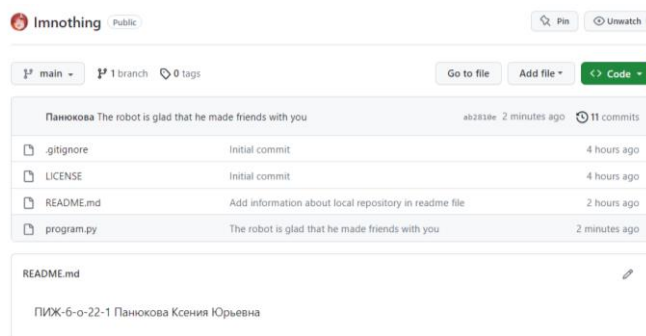


Рисунок 6.5 – Мой репозиторий

## 7. Добавила файл README и зафиксировала сделанные изменения.

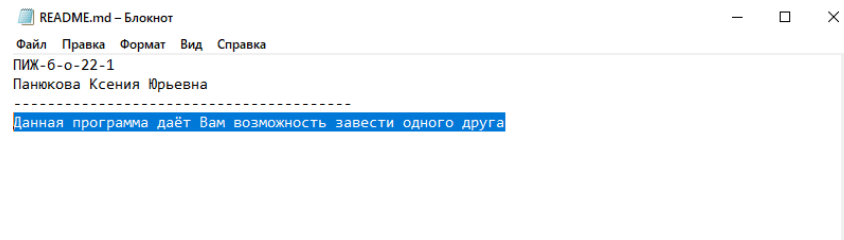


Рисунок 7.1 – Добавляю файл README с описанием программы

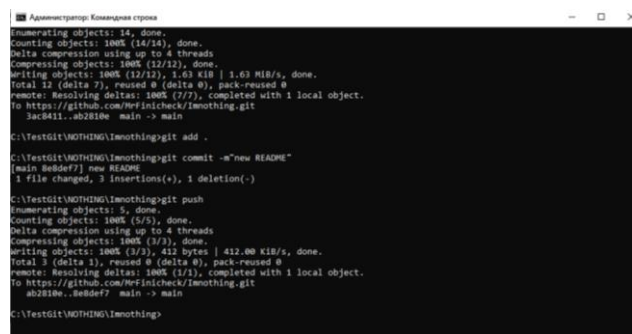


Рисунок 7.2 – Фиксирую сделанные изменения

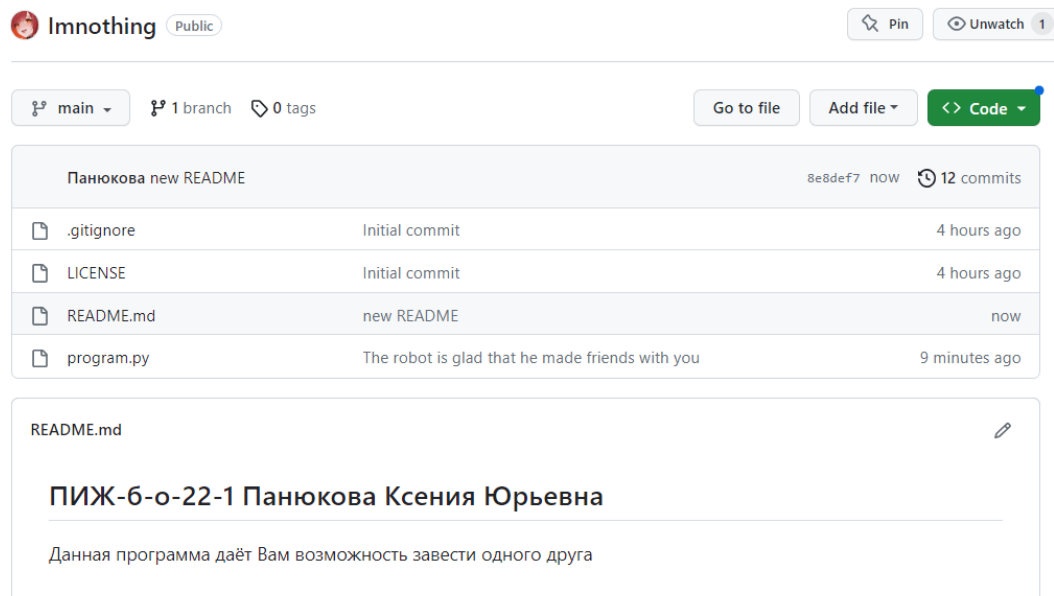


Рисунок 7.3 – Мой репозиторий с измененным файлом README

## Контрольные 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?

В локальных СКВ можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели. В централизованных СКВ самый очевидный минус — это единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками. Если жёсткий диск, на котором хранится центральная БД, повреждён, а своевременные бэкапы отсутствуют, вы потеряете всё.

3. К какой СКВ относится Git?

Распределённая система контроля версий

4. В чем концептуальное отличие Git от других СКВ?

Основное отличие Git от любой другой СКВ — это подход к работе со своими данными.

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.

7. Что такое профиль пользователя в GitHub?

На странице профиля отображаются сведения о вашей работе через репозитории, которые вас интересуют, вклад, который вы сделали, и беседы, в которых вы участвовали.

8. Какие бывают репозитории в GitHub?

Публичные и приватные

9. Укажите основные этапы модели работы с GitHub.

Создание аккаунта, создание репозитория, клонирование репозитория на локальный диск, отправка изменений на GitHub с помощью `git push`.

10. Как осуществляется первоначальная настройка Git после установки?

```
git config --global <user.name>
```

```
git config --global <user.email>
```

11. Опишите этапы создания репозитория в GitHub.

На GitHub в правом верхнем углу, рядом с аватаром есть кнопка с плюсиком, нажимая которую мы переходим к созданию нового репозитория. В результате будет выполнен переход на страницу создания репозитория. Наиболее важными на ней являются следующие поля: Имя репозитория, описание Public/private, `gitignore` и LICENSE. После заполнения этих полей нажимаем кнопку Create repository.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Список лицензий можно просмотреть при создании нового репозитория

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

Для этого на странице репозитория необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования. Затем открыть командную строку или терминал и перейти в каталог, куда вы хотите скопировать хранилище. Затем написать `git clone` и ввести адрес. Это нужно для того, чтобы сохранить наши данные на локальный диск.

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

В командной строке с помощью команды `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 с помощью команды `git add`; фиксации (коммита) изменений с помощью команды `git commit` и отправки изменений на сервер с помощью команды `git push`?

1. В репозитории появляется изменённый, но не подготовленный файл.
2. Появляются изменённые, но подготовленные файлы.
3. После того как выполнен коммит происходит фиксация изменений всех подготовленных файлов, в результате репозиторий не содержит подготовленных файлов. После этого с помощью команды `git push` фиксированные файлы отправляются на GitHub.

#### 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 обоих компьютерах выполняется команда `git clone`. Оба пользователя вносят в программу необходимые изменения, фиксируют их с помощью

команды коммит. Перед командой `push` для синхронизации обоих компьютеров с GitHub выполняют команду `git pull` и только потом `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.

#### Преимущества и недостатки GitLab

##### Преимущества:

- Бесплатный план без ограничений, но есть планы оплаты.
- Это лицензия с открытым исходным кодом.
- Разрешает самостоятельный хостинг на любом плане.
- Он очень хорошо интегрирован с Git.

##### Недостатки:

- Его интерфейс может быть несколько медленнее по сравнению с конкурентами.

#### Преимущества и недостатки GitHub

##### Преимущества:

- Бесплатное обслуживание, хотя есть и платные.
- Очень быстрый поиск в структуре репозитория.
- Большое сообщество и легко найти помощь.
- Он предлагает практические инструменты для сотрудничества и хорошую интеграцию с Git.

- Легко интегрируется с другими сторонними сервисами.

- Он также работает с TFS, HG и SVN.

##### Недостатки:

- У него есть ограничения по пространству, так как вы не можете превышать 100 МБ в одном файле, в то время как репозитории ограничены 1 ГБ в бесплатной версии.

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.

На графическом интерфейсе программы GitHub Desktop выбираем необходимый репозиторий. Указываем название коммита и его описание, затем нажимаем кнопку коммит. Происходит фиксация изменений на локальной машине. После этого нажимаем кнопку push и происходит передача данных на GitHub.