

**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  
«Программирование на Python»**

Выполнил студент группы ИВТ-б-о-22-1

Сумин Никита Сергеевич.

«» \_\_\_\_\_ 2023г.

Подпись студента \_\_\_\_\_

Работа защищена «    » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_

(подпись)

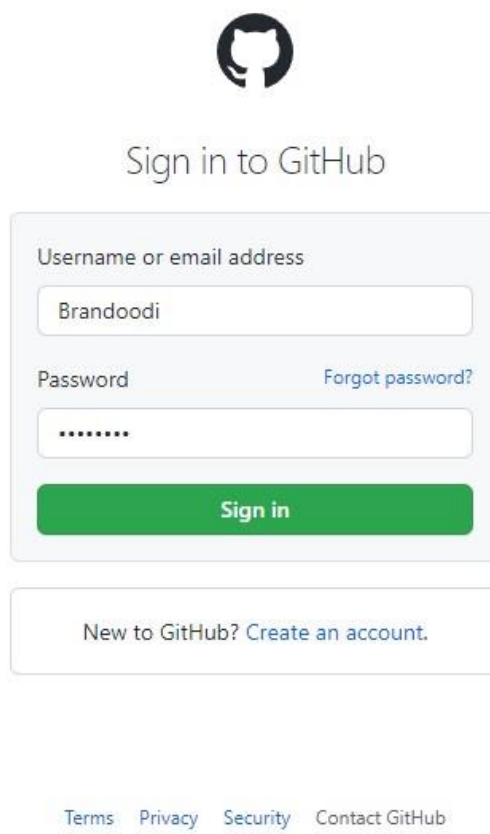
Ставрополь 2023

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

**Порядок выполнения работы:**

**1. 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).**

1. Авторизация на GitHub.

The image shows the GitHub login interface. At the top is the GitHub logo (an octocat). Below it is the text "Sign in to GitHub". The main form has two input fields: "Username or email address" with the text "Brandoodi" and "Password" with masked characters "\*\*\*\*\*". To the right of the password field is a link "Forgot password?". Below the inputs is a green "Sign in" button. At the bottom of the form is a link "New to GitHub? Create an account.". At the very bottom of the page are links for "Terms", "Privacy", "Security", and "Contact GitHub".

Sign in to GitHub

Username or email address

Brandoodi

Password [Forgot password?](#)

\*\*\*\*\*

Sign in

New to GitHub? [Create an account.](#)

[Terms](#) [Privacy](#) [Security](#) [Contact GitHub](#)

Рисунок 1 - Процесс авторизации

2. Создание нового репозитория.

## Create a new repository

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

Owner \* BrandooDi / Repository name \* mywork ✓

Great repository names are short and memorable. Need inspiration? How about [cuddly-winner?](#)

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: None

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

License: None

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

Create repository

Рисунок 2 - Создание репозитория

**2. Выполните клонирование созданного репозитория на рабочий компьютер.**

1. Установка Git.
2. Проверка установки Git.

```
MINGW64:/c/Users/user

user@MINGW64 ~
$ git version
git version 2.36.0.windows.1

user@MINGW64 ~
$
```

Рисунок 3 - Проверка установки 3.

Адрес для клонирования репозитория.

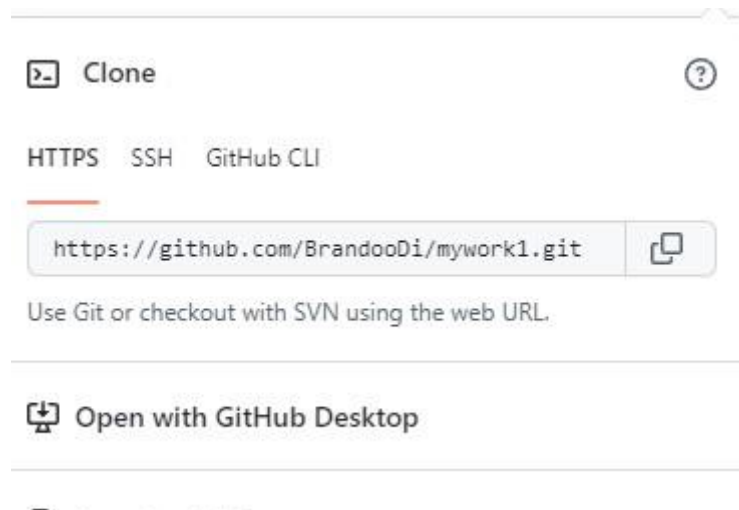


Рисунок 4 - Адрес репозитория 4.

Клонирование репозитория на компьютер.

```
MINGW64:/d/www
user@MINGW64 ~
$ cd:d/www
bash: cd:d/www: No such file or directory

user@MINGW64 ~
$ cd D:www

user@MINGW64 /d/www
$ git clone https://github.com/BrandooDi/mywork1.git
Cloning into 'mywork1'...
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.

user@MINGW64 /d/www
$
```

Рисунок 5 - Клонирование репозитория

3. Дополните файл **.gitignore** необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.

Дополнение файла **.gitignore**:

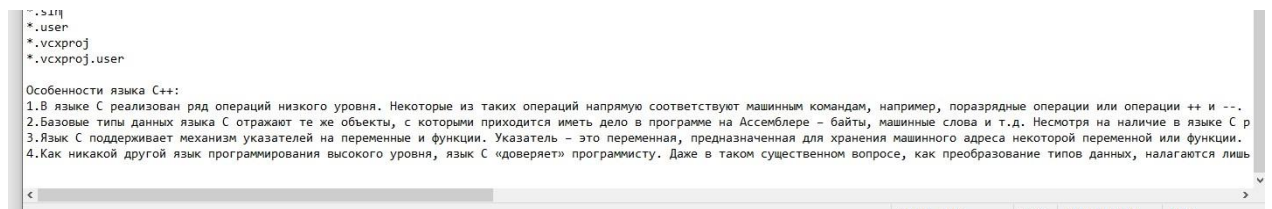


Рисунок 6 - Файл .gitignore

**4. Добавьте в файл README.md информацию о группе и ФИО студента, выполняющего лабораторную работу.**

Добавление информации в файл README.md:

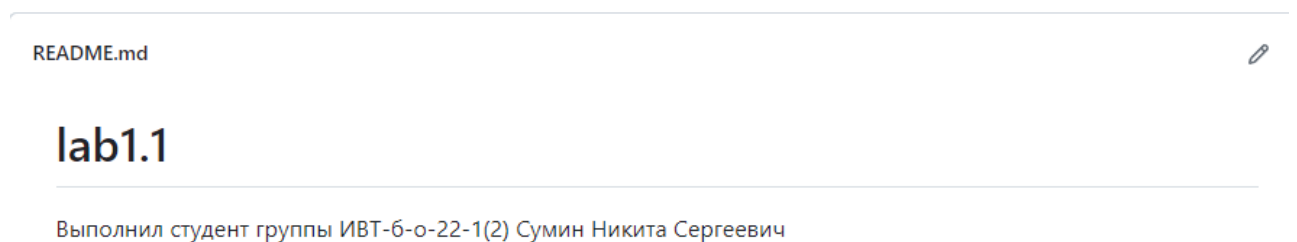
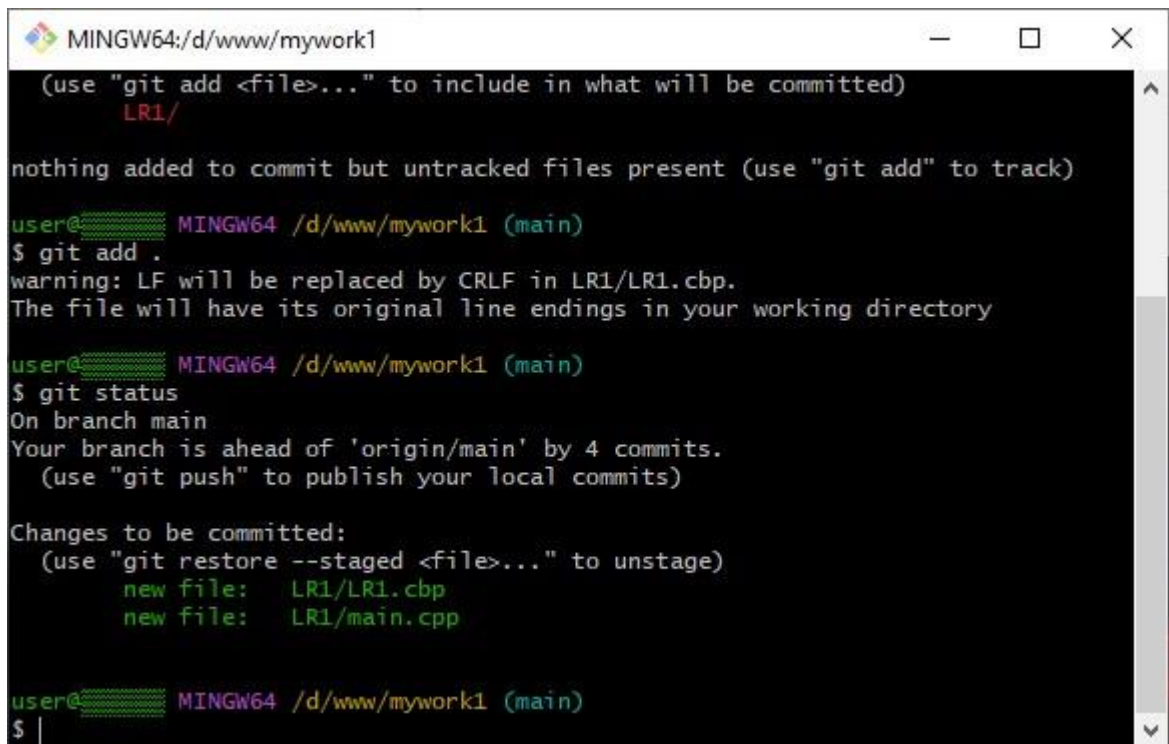


Рисунок 7 - Файл README

**5. Напишите небольшую программу на выбранном Вами языке программирования. Фиксируйте изменения при написании программы в локальном репозитории. Должно быть сделано не менее 7 коммитов.**

Создание файла и добавление его в локальный репозиторий:



```
MINGW64:/d/www/mywork1
(use "git add <file>..." to include in what will be committed)
LR1/

nothing added to commit but untracked files present (use "git add" to track)

user@MINGW64 /d/www/mywork1 (main)
$ git add .
warning: LF will be replaced by CRLF in LR1/LR1.cbp.
The file will have its original line endings in your working directory

user@MINGW64 /d/www/mywork1 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   LR1/LR1.cbp
        new file:   LR1/main.cpp

user@MINGW64 /d/www/mywork1 (main)
$ |
```

Рисунок 8 - Создание файла

Изменение программы и фиксирование 7 коммитов:

```
MINGW64:/d/www/mywork1
user@MINGW64 /d/www/mywork1 (main)
$ git commit -m "1"
[main a5680b3] 1
1 file changed, 1 insertion(+), 1 deletion(-)

user@MINGW64 /d/www/mywork1 (main)
$ git add .

user@MINGW64 /d/www/mywork1 (main)
$ git commit -m "2"
[main a486999] 2
1 file changed, 2 insertions(+)

user@MINGW64 /d/www/mywork1 (main)
$ git add .

user@MINGW64 /d/www/mywork1 (main)
$ git commit -m "3"
[main 33bdb7e] 3
1 file changed, 1 insertion(+)

user@MINGW64 /d/www/mywork1 (main)
$ git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
    add

user@MINGW64 /d/www/mywork1 (main)
$ git add .

user@MINGW64 /d/www/mywork1 (main)
$ git commit -m "4"
[main bf22069] 4
1 file changed, 2 insertions(+)

user@MINGW64 /d/www/mywork1 (main)
$ git add .

user@MINGW64 /d/www/mywork1 (main)
$ git commit -m "5"
[main c7a2c47] 5
1 file changed, 1 insertion(+)

user@MINGW64 /d/www/mywork1 (main)
$ git add .

user@MINGW64 /d/www/mywork1 (main)
$ git commit -m "6"
[main 9cc608d] 6
1 file changed, 2 insertions(+), 1 deletion(-)

user@MINGW64 /d/www/mywork1 (main)
$ git add .

user@MINGW64 /d/www/mywork1 (main)
$ git commit -m
error: switch 'm' requires a value

user@MINGW64 /d/www/mywork1 (main)
$ git commit -m "7"
[main f8b8622] 7
1 file changed, 13 insertions(+), 8 deletions(-)
```

Рисунок 9 - Коммиты 1-7

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      setlocale(0, "");
7      double x, y, z, S;
8
9      cout << "Введите x,y,z\n";
10     cin >> x >> y >> z;
11
12     if (x + y > z)
13     {
14         cout << "s= x+y+z\n";
15     }
16     else {
17         if (x + y <= z)
18             cout << "s= x+y-z.";
19     }
20     return 0;
21 }
22

```

Рисунок 10 - Конечный вид программы

Добавление изменений в репозиторий на Github:

```

MINGW64:/d/www/mywork1

user@MINGW64 ~
$ cd D:/www/mywork1

user@MINGW64 /d/www/mywork1 (main)
$ git pull
Already up to date.

user@MINGW64 /d/www/mywork1 (main)
$ git push
Enumerating objects: 48, done.
Counting objects: 100% (48/48), done.
Delta compression using up to 12 threads
Compressing objects: 100% (47/47), done.
Writing objects: 100% (47/47), 5.24 KiB | 1.75 MiB/s, done.
Total 47 (delta 10), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (10/10), done.
To https://github.com/BrandooDi/mywork1.git
   9361e11..f8b8622  main -> main

user@MINGW64 /d/www/mywork1 (main)
$

```

Рисунок 11 - Функции pull и push

**Ответы на 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?**

*Локальные СКВ:* многие в качестве метода контроля версий применяют копирование файлов в отдельную директорию. Такой подход очень распространён из-за его простоты, однако он невероятно сильно подвержен появлению ошибок. Можно легко забыть, в какой директории находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели.

*Централизованные СКВ:* 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.

## **3. К какой СКВ относится Git?**

Git относится к распределённым системам, поэтому не зависит от центрального сервера, где хранятся файлы.

## **4. В чем концептуальное отличие Git от других СКВ?**

Git не хранит и не обрабатывает данные таким же способом как другие СКВ. Каждый раз, когда вы делаете коммит, т. е. сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Следует, что Git эффективен в хранении бэкапов, поэтому известно мало случаев, когда кто-то терял данные при его использовании.

## **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. Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его основы. В итоге информация не теряется во время её передачи и файл не повредится без ведома Git.

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

*Зафиксированный файл* – файл уже сохранён в вашей локальной базе.

*Измененный файл* – файл, который поменялся, но ещё не был зафиксирован.

*Подготовленный файл* — это изменённый файл, отмеченный для включения в следующий коммит.

## **7. Что такое профиль пользователя в GitHub?**

Профиль – ваша публичная страница на GitHub, как и в социальных сетях. Когда мы ищем работу в качестве программиста, работодатели могут посмотреть наш профиль GitHub и принять его во внимание, когда будут решать, брать нас на работу или нет.

## **8. Какие бывают репозитории в GitHub?**

Репозиторий Git бывает локальный и удалённый.

Локальный репозиторий — это подкаталог `.git`, создаётся (в пустом виде) командой `git init` и (в непустом виде с немедленным копированием содержимого родительского удалённого репозитория и простановкой ссылки на родителя) командой `git clone`. Практически все обычные операции с системой контроля версий, такие, как коммит и слияние, производятся только с локальным репозиторием.

Удалённый доступ к репозиториям Git обеспечивается git- daemon, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Удалённый репозиторий можно только синхронизировать с локальным как «вверх» (*push*), так и «вниз» (*pull*).

## **9. Укажите основные этапы модели работы с GitHub.**

- 1) Регистрация.
- 2) Создание репозитория.
- 3) Клонирование репозитория.

## **10. Как осуществляется первоначальная настройка Git после установки?**

- 1) Убедимся, что Git установлен используя команду: `git version`;
- 2) Перейдём в папку с локальным репозиторием, используя команду: `cd /d`;
- 3) Свяжем локальный репозиторий и удалённый командами:  
`git config --global user.name <YOUR_NAME>`  
`git config --global user.email <EMAIL>`

## **11. Опишите этапы создания репозитория в GitHub.**

- 1) В правом верхнем углу, рядом с аватаром есть кнопка с плюсиком, нажимая которую переходим к созданию нового репозитория.
- 2) В результате будет выполнен переход на страницу создания репозитория. Наиболее важными на ней являются следующие поля:
  - Имя репозитория. Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиториях, которые вы создавали.

- Описание (Description). Можно оставить пустым.
  - Public/private. Выбираем открытый (Public), НЕ ставим галочку “Initialize this repository with a README” (в README потом будет лежать какая-то основная информация, что же такое ваш проект и как с ним работать).
- .gitignore и LICENSE можно сейчас не выбирать. После заполнения этих полей нажимаем кнопку Create repository.

## **12. Какие типы лицензий поддерживаются GitHub при создании репозитория?**

Microsoft Reciprocal License, The Code Project Open License (CPOL), The Common Development and Distribution License (CDDL), The Microsoft Public License (Ms-PL), The Mozilla Public License 1.1 (MPL 1.1), The Common Public License Version 1.0 (CPL), The Eclipse Public License 1.0, The MIT License, The BSD License, The Apache License, Version 2.0, The Creative Commons Attribution-ShareAlike 2.5 License, The zlib/libpng License, A Public Domain dedication, The Creative Commons Attribution 3.0 Unported License, The Creative Commons Attribution-Share Alike 3.0 Unported License, The Creative Commons Attribution-NoDerivatives 3.0 Unported, The GNU Lesser General Public License (LGPLv3), The GNU General Public License (GPLv3).

## **13. Как осуществляется клонирование репозитория GitHub?**

### **Зачем нужно клонировать репозиторий?**

1) После создания репозитория его необходимо клонировать на ваш компьютер. Для этого на странице репозитория необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования.

2) Откройте командную строку или терминал и перейдите в каталог, куда вы хотите скопировать хранилище. Затем напишите `git clone` и введите адрес.

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

Используя команду: `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` ?**

Файлы обновятся на удалённом репозитории.

**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` .

1) Клонировать репозиторий на каждый из компьютеров, используя команду `git clone` и ссылку.

2) Для синхронизации изменений используем команду `git pull`.

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

GitLab — альтернатива GitHub номер один. GitLab предоставляет не только веб-сервис для совместной работы, но и программное обеспечение с открытым исходным кодом.

SourceForge — ещё одна крупная альтернатива GitHub, сконцентрировавшаяся на Open Source. Многие дистрибутивы и приложения Linux обитают на SourceForge

Launchpad — платформа для совместной работы над программным обеспечением от Canonical, компании-разработчика Ubuntu. На ней размещены PPA-репозитории Ubuntu, откуда пользователи загружают приложения и обновления.

**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.**

1) GitHub Desktop это бесплатное приложение с открытым исходным кодом, разработанное GitHub. С его помощью можно взаимодействовать с GitHub, а также с другими платформами (включая GitLab).

2) Fork это весьма продвинутый GUI-клиент для macOS и Windows (с бесплатным пробным периодом). В фокусе этого инструмента скорость, дружелюбность к пользователю и эффективность. К особенностям Fork можно отнести красивый вид, кнопки быстрого доступа, встроенную систему разрешения конфликтов слияния, менеджер репозитория, уведомления GitHub.

3) Sourcetree это бесплатный GUI Git для macOS и Windows. Его

применение упрощает работу с контролем версий и позволяет сфокусироваться на действительно важных задачах.

4) martGit это Git-клиент для Mac, Linux и Windows. Имеет богатый функционал. В арсенале SmartGit вы найдете CLI для Git, графическое отображение слияний и истории коммитов, SSH-клиент, Git-Flow, программу для разрешения конфликтов слияния.

**Вывод:** исследовал базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.