

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Отчет по лабораторной работе № 1.1**

**«Исследование основных возможностей Git и GitHub»**

**по дисциплине «Основы программной инженерии»**

Выполнил студент группы

ПИЖ-б-о-21-1

Разворотников Денис

«10» сентября 2022 г.

Подпись студента \_\_\_\_\_

Работа защищена

« » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

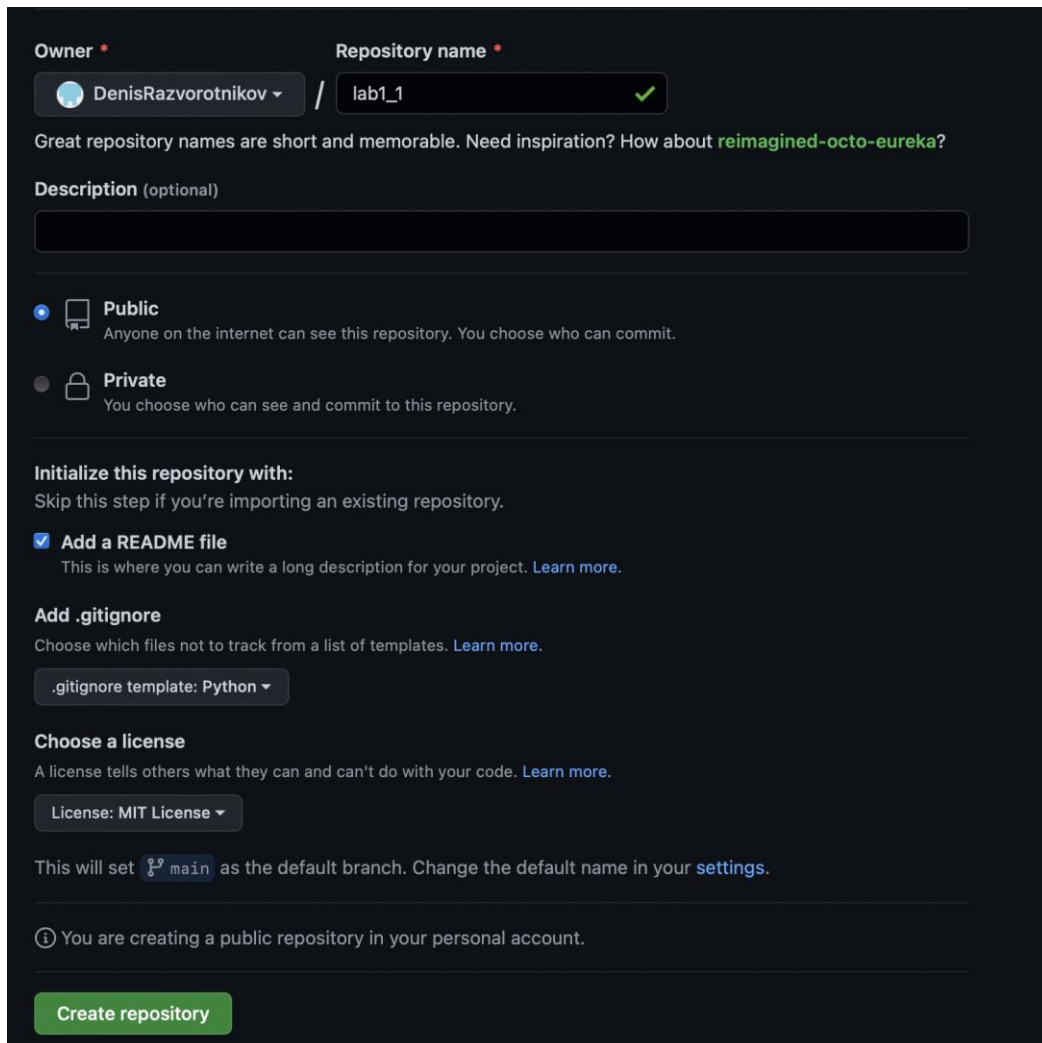
Ставрополь, 2022

## Цель работы:

Исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

## Выполнение работы:

Создание репозитория на GitHub, рисунки 1 и 2.



The screenshot shows the GitHub repository creation interface. At the top, the 'Owner' is set to 'DenisRazvorotnikov' and the 'Repository name' is 'lab1\_1', which is marked as valid with a green checkmark. Below this, a hint suggests repository names should be short and memorable, with an example 'reimagined-octo-eureka?'. The 'Description' field is empty. The 'Public' option is selected, indicating that anyone on the internet can see the repository. The 'Private' option is also visible. Under 'Initialize this repository with:', the 'Add a README file' checkbox is checked. Below this, there are options to 'Add .gitignore' (with a dropdown set to 'Python') and to 'Choose a license' (with a dropdown set to 'MIT License'). A note at the bottom states, 'This will set `main` as the default branch. Change the default name in your settings.' A green 'Create repository' button is at the bottom right.

Owner \* DenisRazvorotnikov / Repository name \* lab1\_1 ✓

Great repository names are short and memorable. Need inspiration? How about [reimagined-octo-eureka?](#)

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: Python

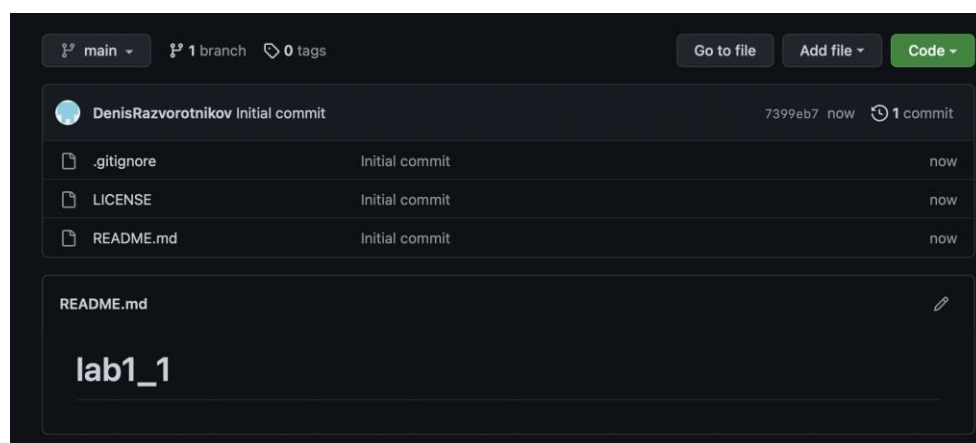
Choose a license  
A license tells others what they can and can't do with your code. [Learn more.](#)  
License: MIT License

This will set `main` as the default branch. Change the default name in your [settings](#).

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

Create repository

Рисунок 1 Страница создания репозитория GitHub



## Рисунок 2 Страница репозитория

Клонирование созданного репозитория на рабочем компьютере (рисунок 3)

```
Cloning into 'lab1_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.
denisrazvorotnikov@MacBook-Air-Denis ~ %
```

## Рисунок 3 Клонирования репозитория

Добавление в файл README.md информации о группе ФИО студента, выполняющего лабораторную работу, (рисунок 4)

```
ПИЖ-Б-0-21-1 Разворотников Денис
```

## Рисунок 4 Окно PyCharm

Написание небольшой программы с фиксацией изменений и созданием коммитов(Рисунок 5 )

> lab1_1 ~/Desktop/lab1_1	1	a=3
> External Libraries	2	b=2
Scratches and Consoles	3	
	4	res=a+b
	5	
	6	print(res)
	7	

## Рисунок 5 Написание программы

```

denisrazvorotnikov@MacBook-Air-Denis lab1_1 % git add .
denisrazvorotnikov@MacBook-Air-Denis lab1_1 % git commit -m "Create prog"
[main 00a9cd5] Create prog
 8 files changed, 97 insertions(+)
 create mode 100644 .idea/.name
 create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
 create mode 100644 .idea/lab1_1.iml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 .idea/workspace.xml
 create mode 100644 prog.py
denisrazvorotnikov@MacBook-Air-Denis lab1_1 %

```

Рисунок 6 Окно терминала

Редактирование файла README.md и фиксация сделанных изменений (Рисунок 7)

Выполнил студент группы ПИЖ-Б-0-21-1 Разворотников Денис

Рисунок 7 – Внесение изменений в файл README

```

denisrazvorotnikov@Air-Denis lab1 % git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .DS_Store
    new file:   .idea/.name
    modified:   .idea/vcs.xml
    new file:   lab1

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
  (commit or discard the untracked or modified content in submodules)
    modified:   README.md
    modified:   lab1 (modified content)

denisrazvorotnikov@Air-Denis lab1 % git add .
denisrazvorotnikov@Air-Denis lab1 % git commit -m "README fix"
[main 140e1d9] README fix
 5 files changed, 4 insertions(+), 1 deletion(-)
 create mode 100644 .DS_Store
 create mode 100644 .idea/.name
 create mode 160000 lab1
denisrazvorotnikov@Air-Denis lab1 %

```

Рисунок 8 Окно терминала

Отправка изменений в локальном репозитории в удаленный репозиторий GitHub


 DenisRazvorotnikov Readme fix		27c797e 1 hour ago	 4 commits
 .idea	Create prog	1 hour ago	
 .gitignore	Initial commit	1 hour ago	
 LICENSE	Initial commit	1 hour ago	
 README.md	Readme fix	1 hour ago	
 prog.py	Create prog	1 hour ago	

Рисунок 9 – Репозиторий после изменений

**Вывод:** При выполнении этой работы была изучена система контроля версий Git и работа с веб-сервисом GitHub

## Контрольные вопросы:

### 1. Что такое СКВ и каково ее назначение?

Система контроля версий — это система, которая регистрирует и отображает изменения в файлах, а также позволяет вернуться к старым версиям этих файлов (откатиться).

### 2. В чем недостатки локальных и централизованных СКВ?

Недостатки локальных СКВ:

- большая вероятность запутаться в копиях (изменить файлы не той версии или заменить не той версией текущую)
- при выходе из строя носителя с файлами при отсутствии бэкапов можно потерять весь проект

Недостатки централизованных СКВ:

- при падении сервера никто не сможет воспользоваться СКВ
- при неисправности носителя с БД при отсутствии бэкапов можно потерять весь проект

### 3. К какой СКВ относится Git?

Git относится к распределённым системам контроля версий. Распределенные СКВ копируют весь репозиторий, что является полной

копией проекта и в случае выхода из строя одного сервера, эту копию можно развернуть на другом без потери данных.

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

Git, в отличие от других СКВ, не хранит информацию об изменениях в каждом файле, а использует систему «снимков». После каждого коммита, система запоминает как выглядит каждый файл и сохраняет ссылку на этот снимок. Если файл не изменился, Git не запоминает снова этот файл, а создает ссылку на предыдущую его версию.

## **5. Как обеспечивается целостность хранимых данных в Git?**

Целостность хранимых данных в Git обеспечивается высчитыванием хэш-сумм вида SHA-1 (40 шестнадцатиричных символов). Эта функциональность встроена в Git на низком уровне. А также, Git сохраняет все объекты в свою базу данных не по имени, а по хеш-сумме содержимого объекта.

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

У файлов в Git есть три состояния:

- **committed** (зафиксированный): файл уже сохранен в локальном репозитории с изменением;
- **modified** (измененный): файл, который изменился, но изменения не были зафиксированы;
- **staged** (подготовленный): измененные файлы, отмеченные для следующего коммита.

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

Профиль пользователя в GitHub – персональная страница программиста, на которой можно увидеть информацию о нём, а также публичные репозитории и его действия в других проектах. Работодатели могут посмотреть и принять во внимание профиль при приёме на работу.

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

Репозитории в профиле GitHub: **public** (публичный), **private** (частный).

Репозитории по месту хранения: локальный репозиторий (копия проекта на ПК), удаленный репозиторий (репозиторий на GitHub).

Также, можно сделать форк репозитория (**origin**) и проводить изменения в нём, не внося изменений в оригинальный (**upstream**).

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

1. Клонирование репозитория локально;
2. Внесение изменений в локальном репозитории;
3. Commit и push в репозиторий GitHub;

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

После установки и проверки работоспособности Git необходимо добавить переменные `user.name` и `user.email` в конфиг. Для этого нужно выполнить следующие команды: `git config --global user.name <YOUR_NAME>` и `git config --global user.email <EMAIL>`.

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

1. Имя репозитория
2. Описание репозитория
3. Вид репозитория: публичный или приватный.
4. Включение/отключение создания файла README.md
5. Выбор файла .gitignore
6. Выбор лицензии

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

Apache License 2.0, GNU General Public License v3.0, MIT License, BSD 2-Clause "Simplified" License, BSD 3-Clause "New" or "Revised" License, Boost Software License 1.0, Creative Commons Zero v1.0 Universal, Eclipse Public License 2.0, GNU Affero General Public License v3.0, GNU General Public License v2.0, GNU Lesser General Public License v2.1, Mozilla Public License 2.0, The Unlicense.

Лицензия MIT — лицензия открытого и свободного программного обеспечения. Является одной из самых ранних свободных лицензий. Она является разрешительной лицензией, то есть позволяет программистам



использовать лицензируемый код в закрытом программном обеспечении при условии, что текст лицензии предоставляется вместе с этим программным обеспечением.

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

Клонирование репозитория осуществляется при помощи Git командой `git clone <link>`. Клонировать репозиторий нужно для хранения локальной копии с возможностью осуществления и применения дальнейших изменений.

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

Командой `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` ?**

Добавление/изменение файла в локальном репозитории Git: `untracked files`.

Добавление нового/ измененного файла под версионный контроль с помощью команды `git add`: `staged`.

Фиксация (коммита) изменений с помощью команды `git commit` и отправки изменений на сервер с помощью команды `git push`: `up to date`.

**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. `git clone <rep. link>`
2. `git add .`
3. `git commit -m "Something new"`
4. `git push`
5. `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.**

Аналоги GitHub: GitLab, BitBucket, Launchpad, GitFlic.

В GitLab разрешение предоставляется на основе ролей людей, в то время как в GitHub разработчики могут предоставлять доступ на чтение или запись к определенным репозиториям. GitLab предлагает бесплатные частные репозитории для проектов с открытым исходным кодом, а GitHub - нет. GitHub способен предоставить полную историю обновлений комментариев - GitLab не поддерживает это. GitLab предоставляет панель мониторинга для анализа времени, планирования и мониторинга.

**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.**

Для GitHub существует приложение с интерфейсом на основе Electron – GitHub Desktop. При первом запуске, приложение потребует входа в учетную запись. Из приложения можно создавать репозитории, клонировать или добавить локальную копию и все это делается в несколько кликов в верхнем

меню программы. Также, при внесении изменений, программа сразу их показывает в том же виде, как и на сайте GitHub.

1. Клонирование репозитория: File -> Clone Repository -> Clone.
2. Изменения автоматически отображаются, добавлять на контроль не нужно.
3. В панели под измененными файлами можно написать описание коммита и закоммитить изменения в ветку.
4. Для пуша в GitHub нужно нажать кнопку Push origin.

