

**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

Логвинов Иван

«10» сентября 2022 г.

Подпись студента _____

Работа защищена

« » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

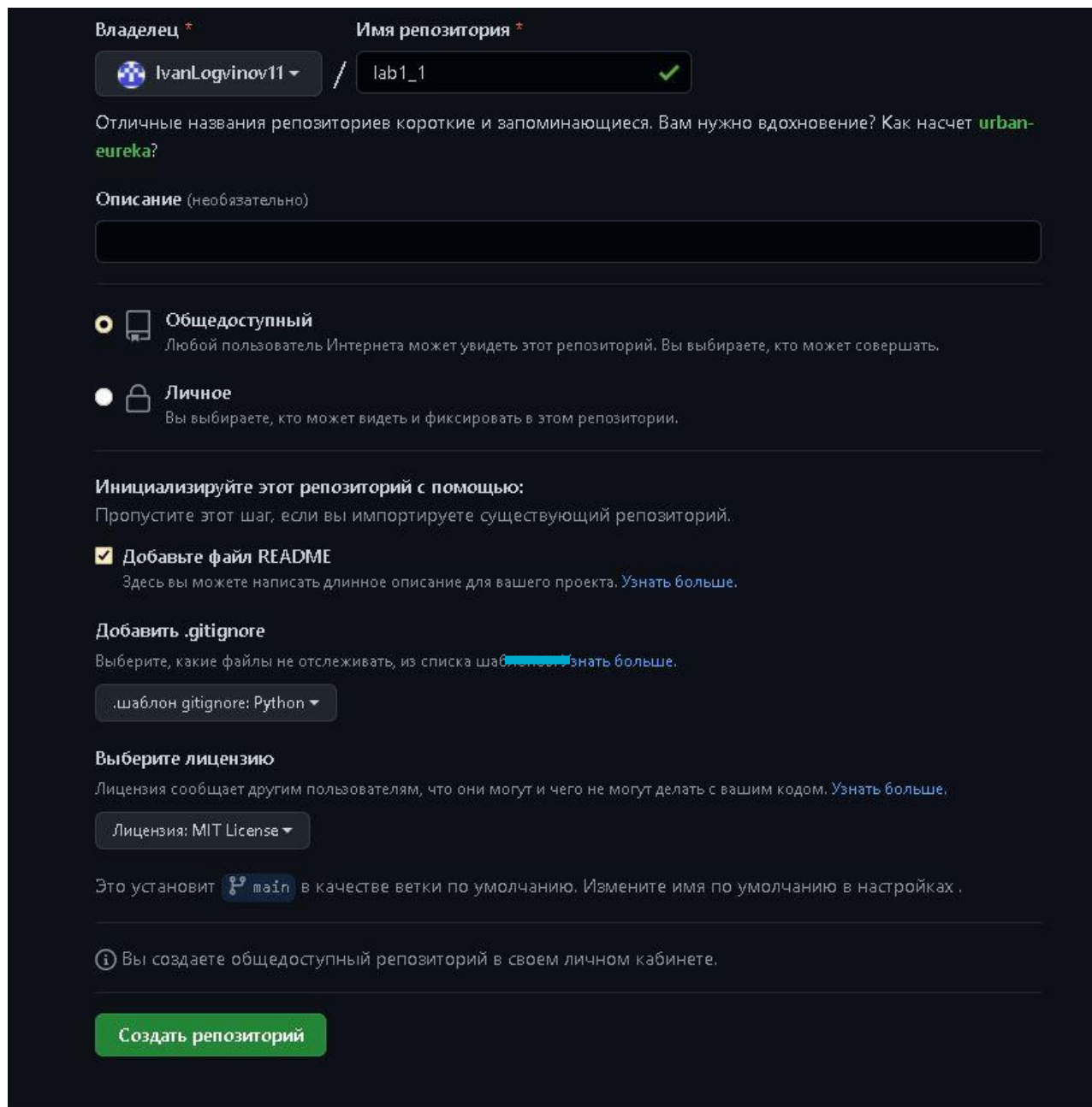
Ставрополь, 2022

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.





Владелец * / Имя репозитория *

IvanLogvinov11 / lab1_1 ✓

Отличные названия репозитория короткие и запоминающиеся. Вам нужно вдохновение? Как насчет **urban-eureka**?

Описание (необязательно)

☒  **Общедоступный**
Любой пользователь Интернета может увидеть этот репозиторий. Вы выбираете, кто может совершать.

☐  **Личное**
Вы выбираете, кто может видеть и фиксировать в этом репозитории.

Инициализируйте этот репозиторий с помощью:
Пропустите этот шаг, если вы импортируете существующий репозиторий.


☒ **Добавьте файл README**
Здесь вы можете написать длинное описание для вашего проекта. [Узнать больше.](#)


Добавить .gitignore
Выберите, какие файлы не отслеживать, из списка шаблонов. [Узнать больше.](#)

.шаблон gitignore: Python ▼

Выберите лицензию
Лицензия сообщает другим пользователям, что они могут и чего не могут делать с вашим кодом. [Узнать больше.](#)

Лицензия: MIT License ▼

Это установит  main в качестве ветки по умолчанию. Измените имя по умолчанию в настройках.

 Вы создаете общедоступный репозиторий в своем личном кабинете.

Создать репозиторий

Рисунок 1 Страница создания репозитория GitHub

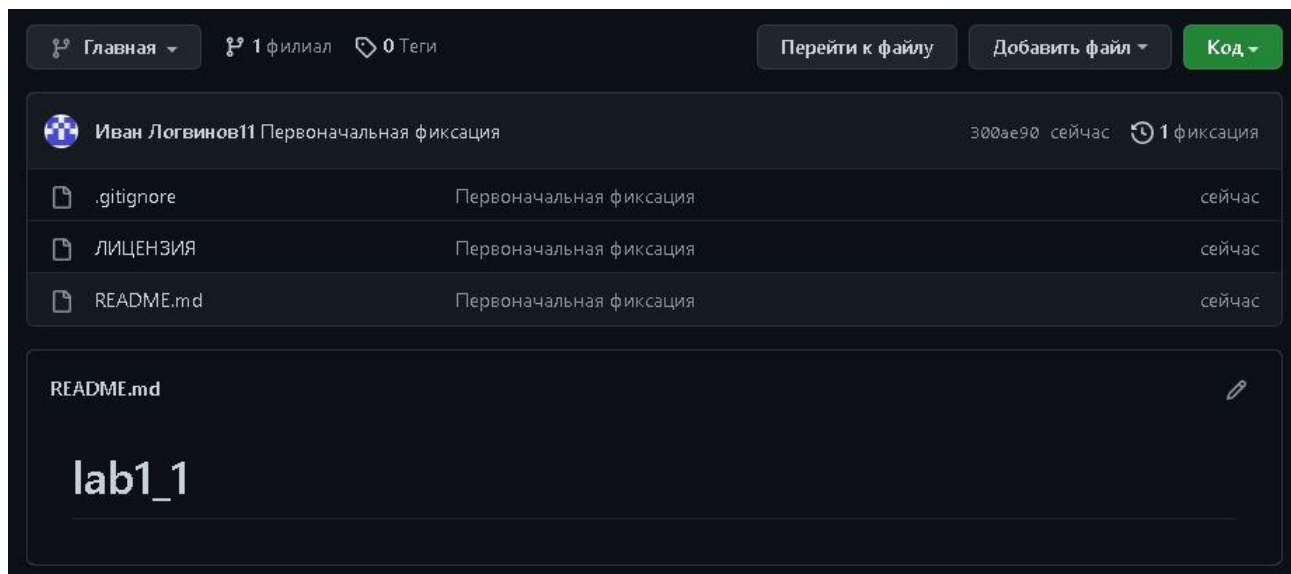


Рисунок 2 Страница репозитория

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.
C:\Users\Иван>
```

Рисунок 3 Клонирования репозитория

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

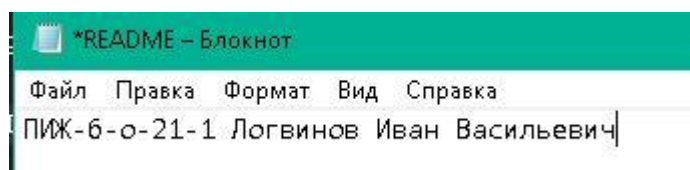


Рисунок 4 Окно PyCharm

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

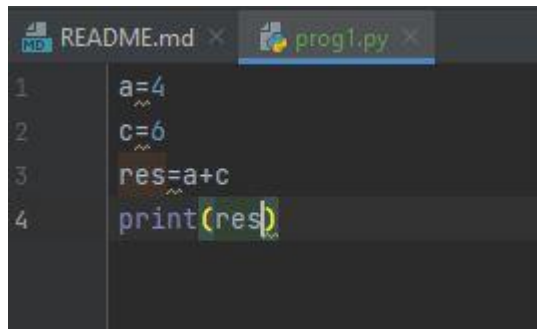


Рисунок 5 Написание программы

```
C:\Users\Иван\Desktop\lab1_1>git add .
warning: in the working copy of '.idea/inspectionProfiles/profiles_settings.xml', LF will be replaced by CR
ime Git touches it

C:\Users\Иван\Desktop\lab1_1>git commit -m "Create work"
[main 7d61617] Create work
8 files changed, 40 insertions(+), 1 deletion(-)
create mode 100644 .idea/.gitignore
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 lab 1.py

C:\Users\Иван\Desktop\lab1_1>
```

Рисунок 6 Окно терминала

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

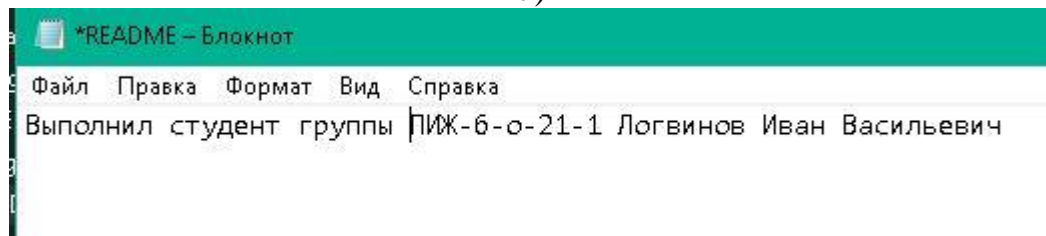


Рисунок 7 – Внесение изменений в файл README

Отправка изменений в локальном репозитории в удаленный репозиторий GitHub

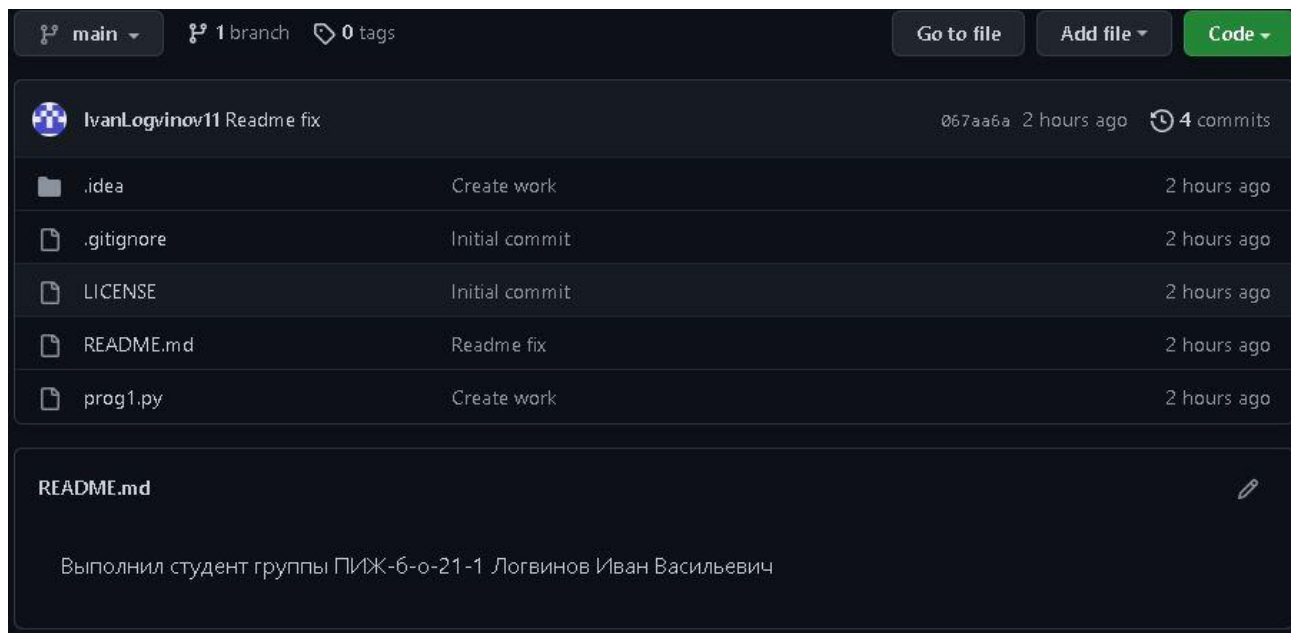


Рисунок 9 – Репозиторий после изменений

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.

