

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

Департамент цифровых, робототехнических систем и электроники

**«Исследование основных возможностей Git и GitHub»
Отчет по лабораторной работе № 1
по дисциплине «Программирование на Python»**

Выполнил студент группы ИВТ-б-о-24-1
Каиров Вадим Сосланович
«__» сентября 2025г.

Подпись студента _____
Работа защищена « » _____ 20__ г.
Проверил Воронкин Р.А. _____
(подпись)

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

Ссылка на GitHub: <https://github.com/ozetin/labochka.git>

Конспект теоретического материала:

Локальные системы контроля версий (СКВ) – это системы, позволяющие отслеживать изменения в файлах и управлять их версиями на одном компьютере, без необходимости подключения к сети или удаленному серверу. Пример такой системы – RCS, которая сохраняла изменения в отдельных файлах.

Централизованные системы контроля версий (ЦСКВ) – системы, где существует один основной сервер с репозиторием кода, к которому подключаются разработчики для получения или отправки изменений. Примеры: CVS – одна из первых ЦСКВ, и более современная Subversion.

Распределённые системы контроля версий (РСКВ) – системы, в которых каждый разработчик имеет полную копию репозитория, включая всю историю изменений. Центральный сервер может использоваться для обмена данными, но не является единственной точкой отказа. Пример: Git. Git отличается от других СКВ своей архитектурой, высокой скоростью работы, удобным управлением ветками и возможностью распределённой работы над проектом.

Хеш-сумма – уникальная цифровая подпись данных, получаемая с помощью специальной хеш-функции. В Git хеш-функции встроены на низком уровне и являются ключевой частью системы. Почти все действия в Git лишь добавляют новые данные в репозиторий, и удалить их или совершить необратимые изменения крайне сложно. До фиксации изменений вы можете потерять или испортить данные, но после коммита восстановить их становится практически невозможно, особенно при регулярной синхронизации с другими репозиториями.

Состояния файлов в Git:

1. Зафиксированное – файл сохранён в локальном репозитории.

2. Изменённое – файл был изменён, но ещё не добавлен в коммит.
3. Подготовленное – изменения отмечены для включения в следующий коммит.

Коммит – это сохранение изменений в репозитории с созданием уникальной точки в истории проекта. Его можно представить как «снимок» состояния всех файлов на конкретный момент.

Git-директория – место хранения метаданных и базы объектов проекта. Это основная часть Git, которая копируется при клонировании репозитория.

Область подготовленных файлов (индекс) – файл в Git-директории, где хранится информация о том, какие изменения будут включены в следующий коммит.

GitHub – платформа для хранения и совместной работы с кодом. В разделе Code обычно находятся два основных файла:

1. README.md – содержит описание проекта; GitHub автоматически отображает его содержимое на странице репозитория.
2. .gitignore – указывает файлы и папки, которые Git должен игнорировать.

Репозиторий может быть публичным, но вносить изменения напрямую могут только соавторы. Остальные пользователи могут предлагать свои изменения через коммиты или пулл-реквесты.

Pull requests – это предложения о внесении изменений в репозиторий. Владельцы проекта могут рассмотреть их и при необходимости интегрировать.

Во вкладке Insight можно найти статистику репозитория, информацию о коммитах и истории изменений

Ход работы:

Для начала был изучен весь предоставленный нам теоретический материал.

Далее создаём репозиторий на GitHub и настраиваем его как нам надо.


Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).
Required fields are marked with an asterisk (*).

1

General

Owner *

 ozetin

Repository name *

labochka

✔ labochka is available.

Great repository names are short and memorable. How about **miniature-octo-computing-machine**?

Description

0 / 350 characters

2

Configuration

Choose visibility *

Choose who can see and commit to this repository

Public

Add README

READMEs can be used as longer descriptions. [About READMEs](#)

Off

Add .gitignore

.gitignore tells git which files not to track. [About ignoring files](#)

Python

Add license

Licenses explain how others can use your code. [About licenses](#)

MIT License

Create repository

Рисунок 1. Создание репозитория.

Клонируем наш репозиторий на рабочий стол

Далее был дополнен файл .gitignore

```
1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
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/
41 .tox/
42 .nox/
43 .coverage
44 .coverage.*
45 .cache
46 nosetests.xml
47 coverage.xml
48 *.cover
49 *.py.cover
50 .hypothesis/
51 .pytest_cache/
```

Рисунок 2. Файл .gitignore

В удалённом репозитории был создан файл и были прописаны в нём ФИО и группа.

Далее была написана программа, по которой было сделано 7 коммитов. После этого они были отправлены в удалённый репозиторий GitHub.

```
C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>git add .

C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>git commit -m "Создал переменную b"
[main 0036d62] Создал переменную b
1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>git add .

C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>git commit -m "Создал переменную c - сумму a и b"
[main 73ab421] Создал переменную c - сумму a и b
1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>git add .

C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>git commit -m "Вывел Введите строку"
[main 4a28754] Вывел Введите строку
1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>git add .

C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>git commit -m "Написал ввод строки"
[main 5d36d0e] Написал ввод строки
1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>git add .

C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>git commit -m "Вывел каждую букву строки"
[main 4054e3d] Вывел каждую букву строки
1 file changed, 3 insertions(+), 1 deletion(-)
```

Рисунок 4. Создание коммитов

```
C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>git add .

C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>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:   .idea/inspectionProfiles/profiles_settings.xml
    new file:   .idea/labochka.iml
    new file:   .idea/misc.xml
    new file:   .idea/modules.xml
    new file:   .idea/vcs.xml
    new file:   main.py

C:\Users\vadim\OneDrive\Рабочий стол\Лаба1\labochka>
```

Рисунок 5. Сохранение изменений на GitHub.

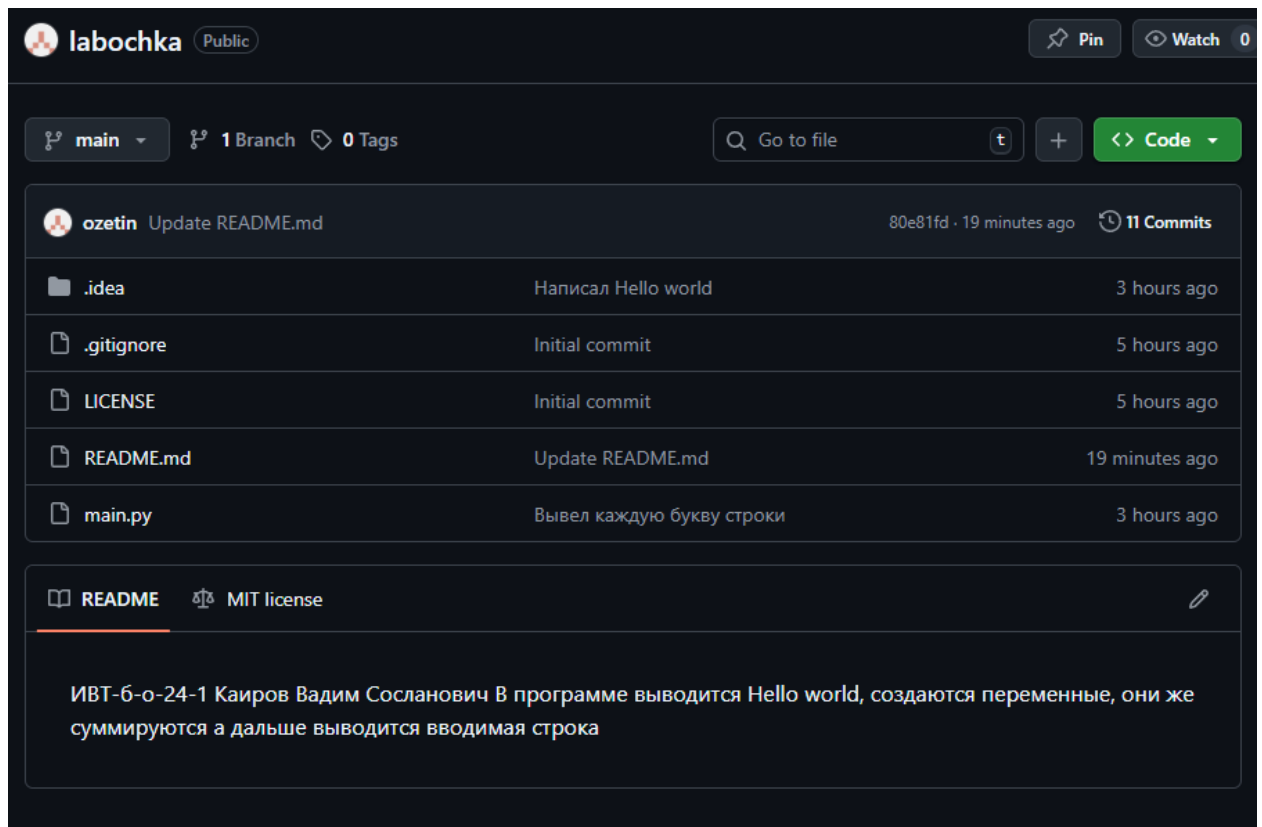


Рисунок 6. Результат лабораторной работы на GitHub.

Контрольные вопросы:

1. Что такое СКВ и какого её назначение?

Это программное обеспечение, которое помогает разработчикам отслеживать и управлять изменениями в исходном коде и других файлах проекта.

2. В чём недостатки локальных и централизованных СКВ?

Недостатки локальных СКВ: Отсутствие совместной работы, резервного копирования.

Недостатки централизованных СКВ: зависимость от сервера, ограниченная работа офлайн.

3. К какой СКВ относится Git?

Git относится к распределённым СКВ.

4. В чём концептуальное отличие Git от других СКВ?

1) Распределённая архитектура. Каждый разработчик имеет полную копию всего репозитория со всей историей изменений.

2) Хранение данных. В Git данные хранятся не как последовательность изменений, а как снимки(снимки состояния файлов в момент коммита). Каждый коммит – это снимок всего проекта.

3) Работа офлайн. Можно выполнять большинство операций полностью офлайн, так как вся история локально доступна.

5. Как обеспечивается целостность хранимых данных в Git?

Каждый объект идентифицируется уникальным хэшем, который вычисляется на основе содержимого этого объекта. Это гарантирует, что любые изменения данных будут обнаружены, так как хэш не совпадёт. Коммиты содержат ссылки на родительские коммиты через их хэши. Структуры каталогов и файлы тоже связаны через хэши, вся история и структура репозитория связаны цепочкой хэшей, что предотвращает подделку или повреждение данных. При чтении или записи объектов Git проверяет их хэши. Если хэш не совпадёт, то Git выдаст ошибку.

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

- 1) Зафиксированное – файл сохранён в локальном репозитории.
- 2) Изменённое – файл был изменён, но ещё не добавлен в коммит.
- 3) Подготовленное – изменения отмечены для включения в следующий коммит.

Таким образом, эти состояния отражают жизненный цикл изменений файла в Git: от сохранённого состояния, через внесённые изменения, до подготовки и фиксации этих изменений в репозитории.

7. Что такое профиль пользователя в GitHub?

Профиль пользователя в GitHub — это персональная страница, которая представляет конкретного пользователя на платформе GitHub. Он содержит информацию о пользователе и его активности в экосистеме GitHub.

8. Какие бывают репозитории в GitHub?

- 1) Публичные. Все могут просматривать, клонировать, форкать такой репозиторий.

2) Приватные. Доступ имеют только выбранные пользователи.

9. Укажите основные этапы модели работы в GitHub.

Создаётся репозиторий на GitHub. Для разработки новой функции или исправления ошибки создаётся отдельная ветка чтобы изолировать изменения от основной ветки. Вносятся изменения на локальной машине. Подготавливаются изменённые файлы в индекс с помощью `git add`, чтобы подготовить их к коммиту. Создаётся коммит с описанием изменений. Отправляется локальный коммит в удалённый репозиторий на GitHub. Отправляется Pull Request. Другие разработчики проверяют изменения. После одобрения изменения добавляются в основную ветку.

10. Как осуществляется первоначальная настройка Git после установки?

Первоначально настраиваются имя пользователя и электронная почта.

`git config --global user.name` и `git config --global user.email`

11. Опишите этапы создания репозитория в GitHub.

Входим в аккаунт на GitHub. Нажимаем на кнопку New repository. Заполняем информацию о репозитории.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

1) MIT License. Позволяет делать с кодом почти всё: использовать, изменять, распространять, даже в коммерческих целях. Единственное условие — сохранить уведомление об авторстве.

2) Apache License 2.0. Похожая на MIT, но дополнительно защищает от патентных претензий. Требуется сохранять уведомления и лицензии при распространении.

3) GPL (General Public License). Строгая лицензия, которая требует, чтобы все производные работы тоже были открыты под GPL. Если вы используете GPL-код, ваш проект тоже должен быть открытым.

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

`git clone <URL_репозитория>`

Работа с кодом локально — можно редактировать, тестировать и коммитить изменения на своём компьютере. Синхронизация с удалённым репозиторием — вы можете получать обновления и отправлять свои изменения обратно на GitHub.

14. Как проверить состояние локального репозитория Git?

`git status` показывает какие файлы изменены, но еще не сохранены

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) Добавление/изменение файла в локальном репозитории

Эти изменения не отслеживаются или помечены как изменённые, но не подготовленные к коммиту. Изменения находятся в `working directory`, но еще не добавлены в индекс.

2) Добавление нового/изменённого файла под версионный контроль с помощью команды `git add`

Команда `git add` добавляет указанные файлы в индекс. Файлы теперь готовы к коммиту.

3) Фиксация (коммит) изменений с помощью команды `git commit`

Команда `git commit` сохраняет состояние файлов из `staging area` в локальный репозиторий. Создаётся новый коммит, который фиксирует изменения. Индекс становится пустым (готов к новым изменениям). Рабочая директория соответствует состоянию последнего коммита.

4) Отправка изменений на сервер с помощью команды `git push`

Команда `git push` отправляет твои локальные коммиты в удалённый репозиторий. Локальный репозиторий не меняется. Изменения становятся доступны на удалённом сервере.

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 ПК1 клонируем репозиторий из GitHub `git clone`, изменяем локальный репозиторий как нам надо и отправляем его в удалённый репозиторий `git push origin main`. На ПК2 клонируем новый репозиторий из GitHub. На ПК2 обновляем локальный репозиторий `git pull origin main`. Аналогично если изменения вносятся на локальный репозиторий ПК2.

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

Существуют также GitLab, Bitbucket, SourceForge.

Сравнение GitHub и GitLab.

1) Тип сервиса

GitHub – облачный сервис

GitLab – облачный + возможность установить на собственный сервер.

2) Управление проектами

GitHub – основной фокус на Git, базовое управление задачами

GitLab - Полноценный CI/CD, управление задачами, доски Kanban, DevOps-инструменты

GitHub подходит для открытых проектов и сотрудничества с огромным сообществом, а также для пользователей, которым нужна простота и мощные интеграции.

GitLab больше ориентирован на полный цикл DevOps: от написания кода до автоматической доставки, и подходит как облачный сервис, так и для корпоративного self-hosting.

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.

Графические клиенты: GitHub Desktop, Sourcetree, GitKraken.

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