

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»

Выполнил:
Матвеев Александр Иванович
1 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

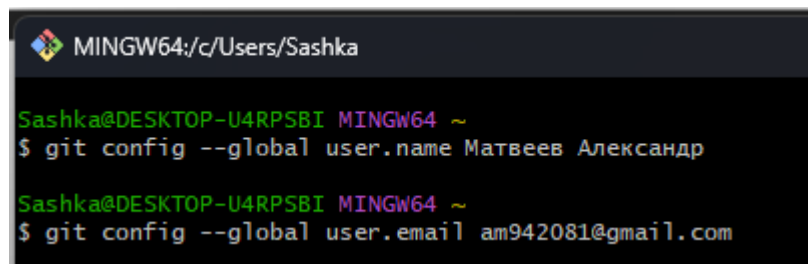
Ставрополь, 2023 г.

Тема: Исследование основных возможностей Git и GitHub.

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

Ход работы.

1. Добавил в настройки Git имя, фамилию и адрес электронной почты, связанный с учетной записью GitHub.

A screenshot of a terminal window with a dark background. The title bar at the top shows a Windows logo icon followed by the text 'MINGW64; c:/Users/Sashka'. The terminal content shows two lines of commands being executed. The first line shows the prompt 'Sashka@DESKTOP-U4RPSBI MINGW64 ~' followed by '\$ git config --global user.name Матвеев Александр'. The second line shows the prompt 'Sashka@DESKTOP-U4RPSBI MINGW64 ~' followed by '\$ git config --global user.email am942081@gmail.com'.

```
MINGW64; c:/Users/Sashka

Sashka@DESKTOP-U4RPSBI MINGW64 ~
$ git config --global user.name Матвеев Александр

Sashka@DESKTOP-U4RPSBI MINGW64 ~
$ git config --global user.email am942081@gmail.com
```

Рисунок 1 – Установка в Git имени и фамилии, а также адреса электронной почты.

2. Создал репозиторий GitHub.

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 * SashkaHacker / Repository name * software_engineering
✓ software_engineering is available.

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

Description (optional)
Это учебный проект по дисциплине: основы программной инженерии.

☒ **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: None

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

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

[Create repository](#)

Рисунок 2 – Создание репозитория GitHub

3. Клонировал репозиторий на компьютер.

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/Desktop
$ git clone https://github.com/SashkaHacker/software_engineering.git
Cloning into 'software_engineering'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

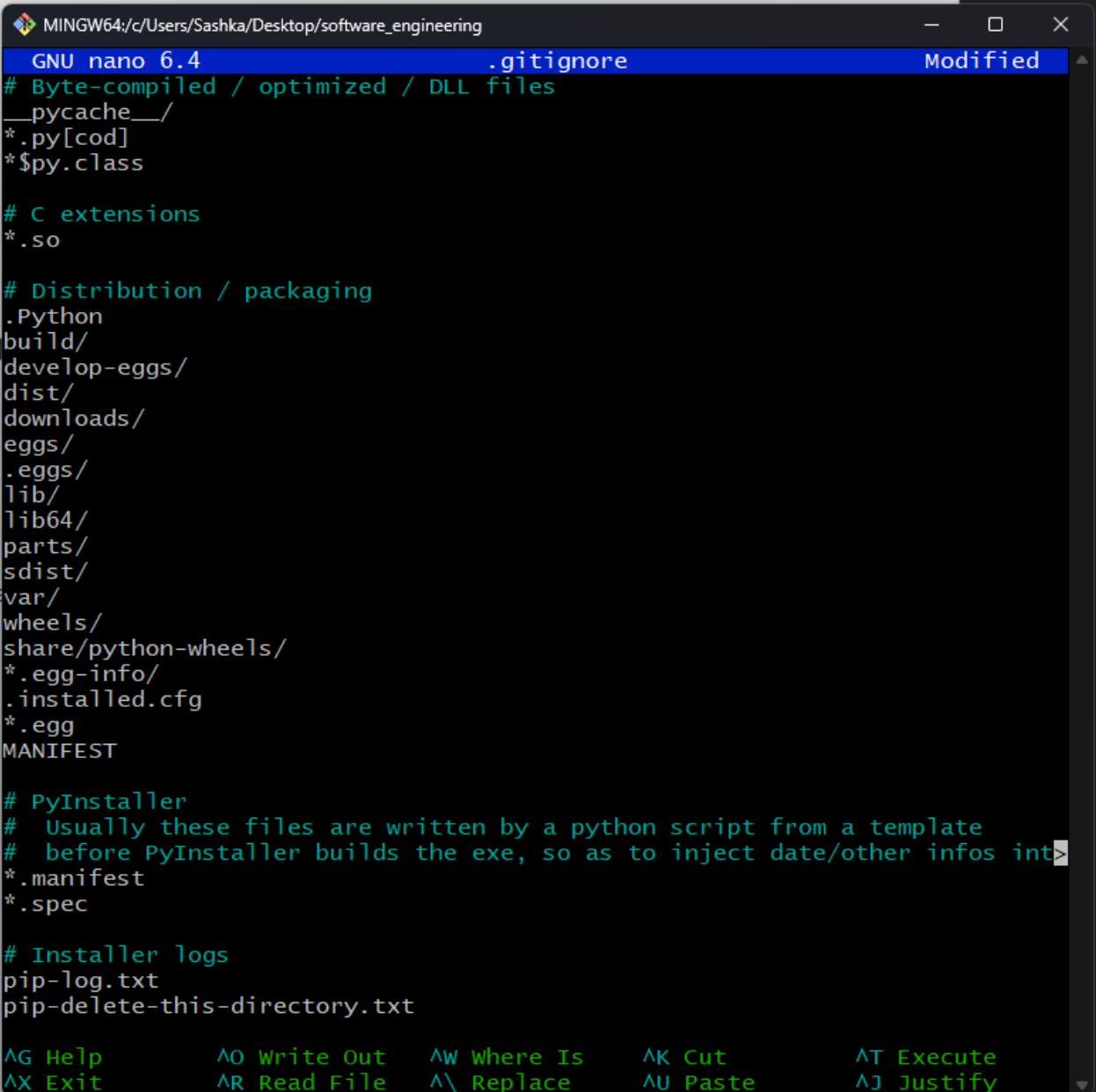
Рисунок 3. – Клонирование репозитория

4. Создал файл README.md с необходимой информацией.

```
MINGW64/c/Users/Sashka/Desktop/software_engineering
GNU nano 6.4 README.md
#1
Студент группы ПИЖ-6-о-22-1, Матвеев Александр.
```

Рисунок 4. – Создание README.md

5. Дополнил файл .gitignore правилами для языка Python.



```
MINGW64:/c/Users/Sashka/Desktop/software_engineering
GNU nano 6.4 .gitignore Modified
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST

# PyInstaller
# Usually these files are written by a python script from a template
# before PyInstaller builds the exe, so as to inject date/other infos into
*.manifest
*.spec

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

Рисунок 5. – Правила в .gitignore

6. Создал программу «Калькулятор», в ходе которой создавал коммиты в локальном репозитории. В

few changes	main	Матвеев	3 minutes ago
add starting programm when starting script		Матвеев	8 minutes ago
add func update		Матвеев	9 minutes ago
add func logicalc		Матвеев	10 minutes ago
add func build		Матвеев	10 minutes ago
add class Main()		Матвеев	11 minutes ago
Creating file "calculator"		Матвеев	13 minutes ago
Initial commit	origin/main	SashkaHacker*	58 minutes ago

Рисунок 6. – Список коммитов

7. Отредактировал README.md файл, учитывая функционал написанной программы.

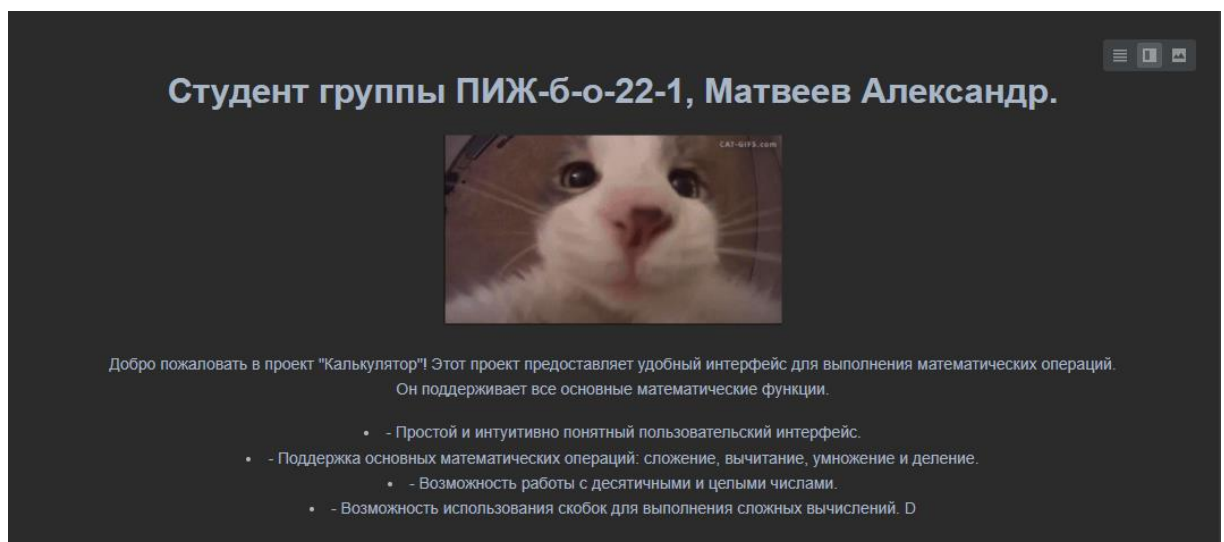


Рисунок 7. – Файл README.md

8. Добавил отчет в репозиторий.

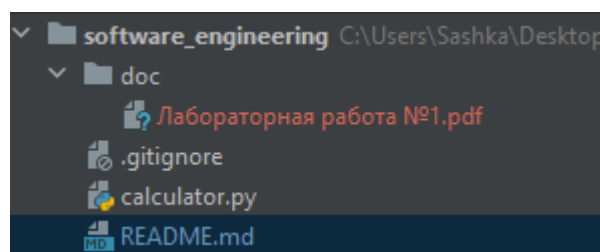


Рисунок 8. – Отчет в локальном репозитории

9. Отправил изменения локального репозитория в удаленный репозиторий GitHub.

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/Desktop/software_engineering (main)
$ git push
```

Рисунок 9. – Отправка изменений на сервер

Контрольные вопросы:

1. В: Что такое СКВ и каково ее назначение?

О: Система контроля версий (СКВ) - это инструмент, используемый для управления изменениями в файловой системе проекта. Ее основное назначение состоит в отслеживании и управлении различными версиями файлов, записывая изменения, делаемые разными людьми и в разное время.

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

О: Для локальных СКВ – это возможность появления ошибок, ведь можно легко забыть, в какой директории вы находитесь и изменить не те файлы. Для центральных СКВ – это возможность утери данных с сервера, ведь там используется один единственный сервер, который содержит все версии файлов.

3. В: К какой СКВ относится Git?

О: Git относится к распределенным системам контроля версий.

4. В: В чем концептуальное отличие Git от других СКВ?

О: Большинство СКВ хранят информацию в виде списка изменений в файлах, а Git при каждом коммите сохраняет снимок всего проекта на данный момент, если в каких то файлах изменений нет, то он оставляет ссылку на предыдущую версию файла.

5. В: Как обеспечивается целостность хранимых данных в Git?

О: 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: В каких состояниях могут находиться файлы в Git? Как связаны эти состояния?

О: У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged). Если версия файла изменена и добавлена в

индекс, значит, она подготовлена. И если файл был изменён с момента последнего распаковывания из репозитория, но не был добавлен в индекс, он считается изменённым.

7. В: Что такое профиль пользователя в GitHub?

О: Профиль пользователя в GitHub — это публичная страница, где отображается информация о пользователе и его активности на платформе GitHub. Профиль является центральным местом, где пользователь может представить себя и свою работу сообществу разработчиков.

8. В: Какие бывают репозитории в GitHub?

О: Публичные и закрытые.

9. В: Укажите основные этапы модели работы с GitHub.

О: Стандартный способ - создать локальный клон удаленного репозитория и работать с ним локально, периодически внося изменения в удаленный репозиторий.

10. В: Как осуществляется первоначальная настройка Git после установки?

О: Указание имени, фамилии, а также почты с GitHub.

11. В: Опишите этапы создания репозитория в GitHub.

О: Нажать на кнопку создания репозитория, дать имя репозиторию, выбрать видимость репозитория, если необходимо отметить флажки создания файлов README и .gitignore.

12. В: Какие типы лицензий поддерживаются GitHub при создании репозитория?

О: Apache license 2.0, Boost Software License 1.0, BSD 2-clause "Simplified" license, BSD 3-clause "New" or "Revised" license, Creative Commons Zero v1.0, Eclipse Public License 2.0, GNU Affero General Public License v3.0, Mozilla Public License 2.0.

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

О: Для клонирования необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования.

Далее необходимо открыть командную строку или терминал и перейти в каталог, куда необходимо скопировать хранилище. Затем написать `git clone` и ввести адрес. Клонирование репозитория GitHub позволяет сохранить локальную копию проекта, работать с кодом, получать изменения и сотрудничать с другими разработчиками.

14. В: Как проверить состояние локального репозитория Git?

О: Использовать команду в терминале: `git status`.

15. В: 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`?

О: - Добавление/изменение файла в локальный репозиторий Git: Файлы, которые были созданы или изменены, переходят в состояние "изменено" (modified). Git отслеживает изменения в файлах, но они еще не были зафиксированы (committed).

- Файлы, которые были добавлены в индекс с помощью команды ``git add``, переходят в состояние "отслеживаемый" (tracked). Git начинает отслеживать изменения в добавленных файлах и готовит их к фиксации (commit).

- Фиксация (коммит) изменений с помощью команды `git commit`: Фиксация изменений с помощью команды ``git commit`` сохраняет состояние всех отслеживаемых файлов в локальном репозитории. Изменения получают уникальный идентификатор, называемый хэш коммита. После коммита файлы переходят в состояние "зафиксировано" (committed).

- Отправка изменений на сервер с помощью команды `git push`: Команда ``git push`` отправляет фиксированные коммиты из локального репозитория на серверный репозиторий. При успешной отправке изменений на сервер, локальный и серверный репозитории станут синхронизированными.

Другие разработчики могут получить доступ к этим изменениям и просмотреть/получить последнюю версию кода.

16. В: 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`.

О: Сначала на первом и втором компьютере необходимо выполнить команду `git clone`. После этого на обоих компьютерах есть локальные копии репозитория, при этом потребуется регулярно синхронизировать изменения между компьютерами и репозиторием GitHub. Для этого необходимо добавить все изменения в локальном репозитории с помощью команды `git add`, затем ввести команду `git commit` для фиксации изменений, после для отправки изменений на GitHub использовать команду `git push`. Чтобы загрузить последние изменения необходимо использовать `git pull`.

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

О: GitLab, Bitbucket, SourceForge. GitHub является крупнейшим онлайн-сообществом разработчиков. GitLab можно развернуть на своем сервере.

18. В: 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.

О: GitHub Desktop, IDE Pycharm, IDE IntelliJ IDEA. Можно создавать новые репозитории, клонировать существующие, отслеживать изменения, создавать ветки, коммиты и многое другое.

