

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:
Соколов Михаил Романович
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., доцент кафедры
инфокоммуникаций

(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 выполнения работы:

1. Создать общедоступный репозиторий на GitHub с использованием лицензии MIT и выбранным языком программирования:

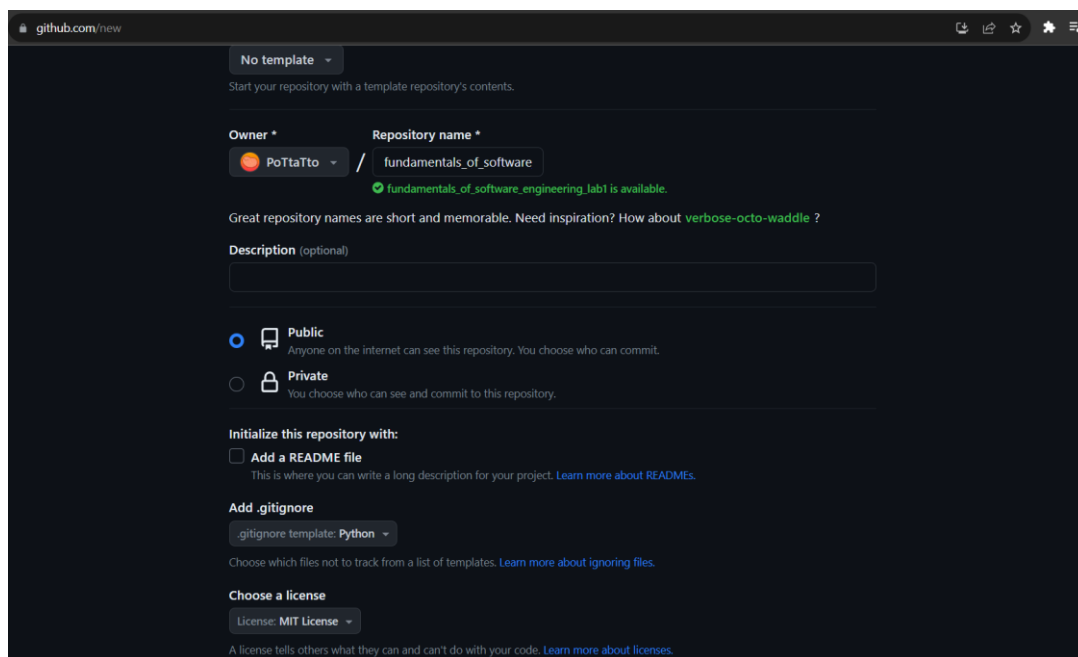


Рисунок 1 – Создание общедоступного репозитория на GitHub с заданными настройками

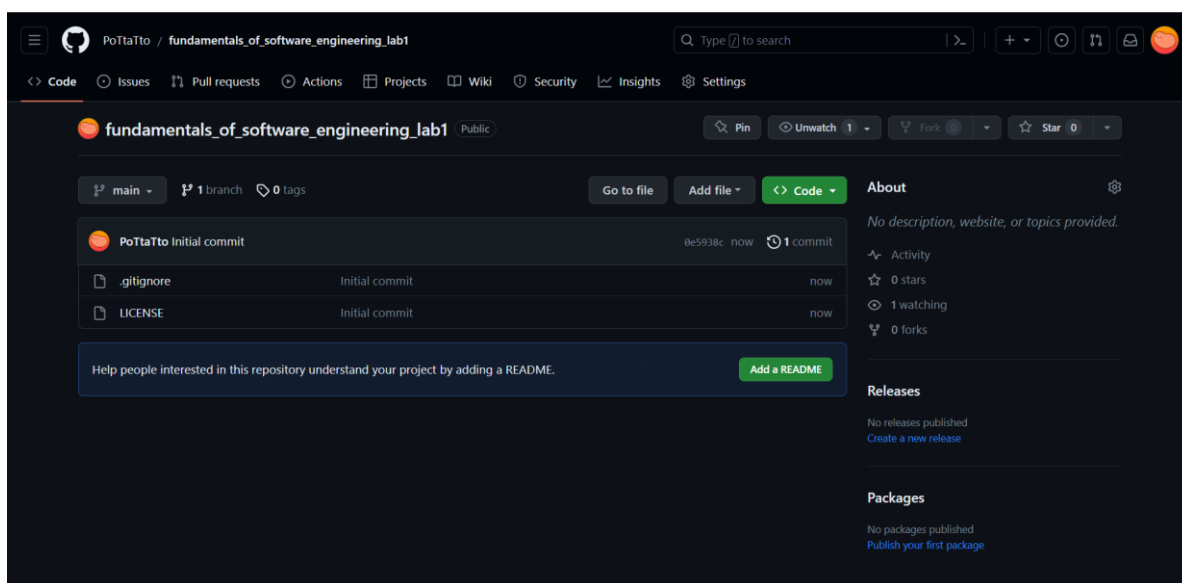
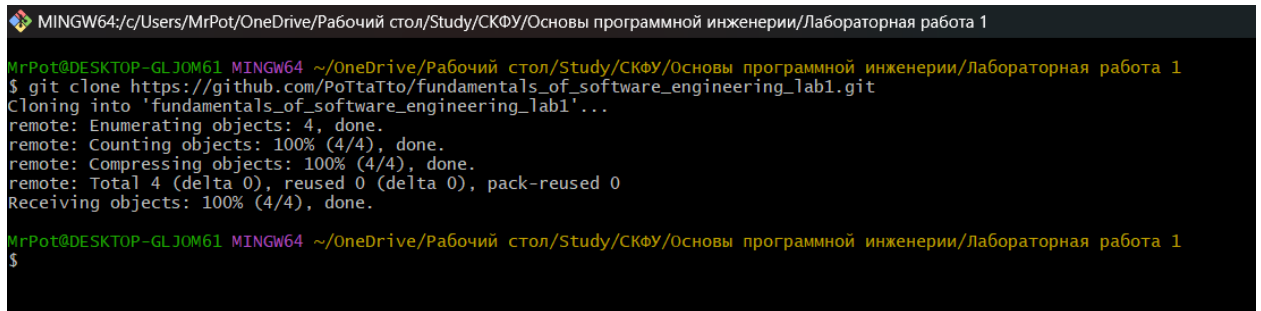


Рисунок 2 – Результат создания репозитория

2. Клонировать репозиторий на рабочий компьютер:

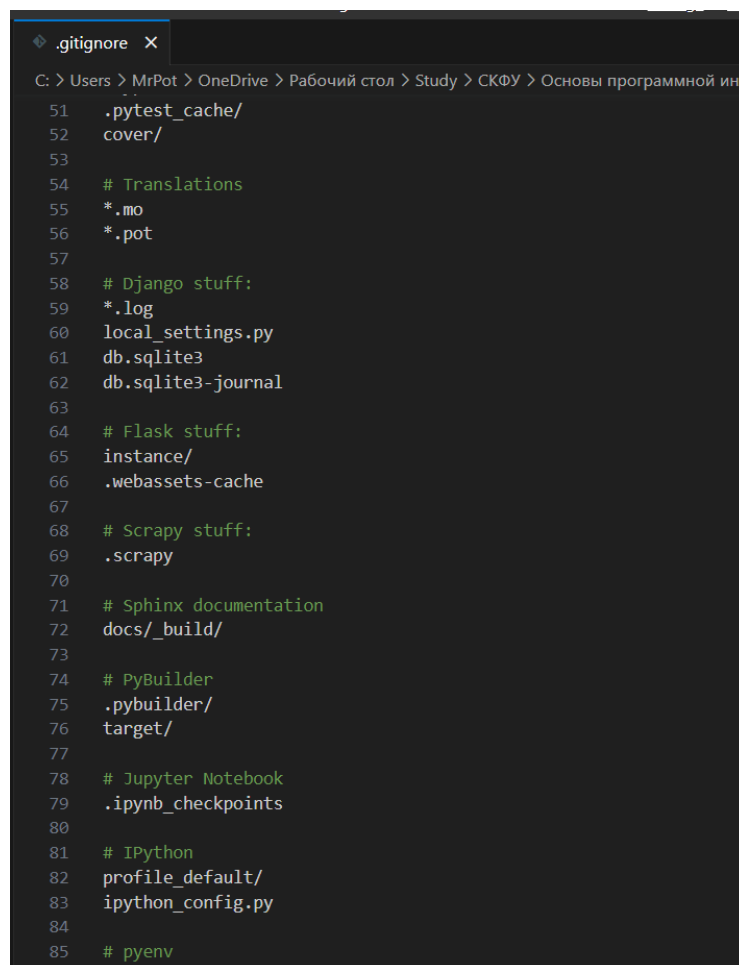


```
MINGW64:/c:/Users/MrPot/OneDrive/Рабочий стол/Study/СКФУ/Основы программной инженерии/Лабораторная работа 1
MrPot@DESKTOP-GLJOM61 MINGW64 ~/OneDrive/Рабочий стол/Study/СКФУ/Основы программной инженерии/Лабораторная работа 1
$ git clone https://github.com/PoTtaTto/fundamentals_of_software_engineering_lab1.git
Cloning into 'fundamentals_of_software_engineering_lab1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
MrPot@DESKTOP-GLJOM61 MINGW64 ~/OneDrive/Рабочий стол/Study/СКФУ/Основы программной инженерии/Лабораторная работа 1
$
```

Рисунок 3 – Клонирование созданного репозитория на локальный компьютер

3. Дополнить файл «.gitignore» необходимыми правилами для выбранного языка:

При создании репозитория и выборе языка Python, GitHub автоматический создал «.gitignore» с нужными правилами игнорирования файлов Python и IDE.



```
.gitignore X
C: > Users > MrPot > OneDrive > Рабочий стол > Study > СКФУ > Основы программной инж
51 .pytest_cache/
52 cover/
53
54 # Translations
55 *.mo
56 *.pot
57
58 # Django stuff:
59 *.log
60 local_settings.py
61 db.sqlite3
62 db.sqlite3-journal
63
64 # Flask stuff:
65 instance/
66 .webassets-cache
67
68 # Scrapy stuff:
69 .scrapy
70
71 # Sphinx documentation
72 docs/_build/
73
74 # PyBuilder
75 .pybuilder/
76 target/
77
78 # Jupyter Notebook
79 .ipynb_checkpoints
80
81 # IPython
82 profile_default/
83 ipython_config.py
84
85 # pyenv
```

Рисунок 4 – Часть созданных, игнорируемых файлов и расширений для Python и IDE

4. Добавить в файл README.md информацию о группе и ФИО автора лабораторной работы:

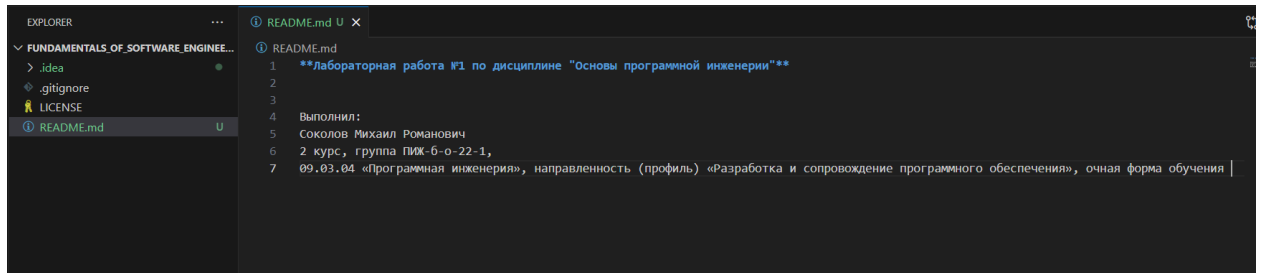


Рисунок 5 – Добавление соответствующей информации в README.md

5. Написать небольшую программу на Python. Фиксировать изменения при написании программы в локальном репозитории:

```
commit f7c5473a2979991283f0ec9e979148eb0a229c5f (HEAD -> main)
Author: PoTtaTto <MrPotatsus@yandex.ru>
Date: Mon Sep 11 17:36:34 2023 +0300

    Модифицирование функции вывода сгенерированного пароля

commit 0e31106e7a2a28807bac4c2ba9c5b671845ee553
Author: PoTtaTto <MrPotatsus@yandex.ru>
Date: Mon Sep 11 17:33:30 2023 +0300

    Добавление функции вывода сгенерированного пароля

commit 2ba071ca8de7464299da4f9801102c0c345c8d83
Author: PoTtaTto <MrPotatsus@yandex.ru>
Date: Mon Sep 11 16:56:48 2023 +0300

    Добавление функции вывода случайной буквы ASCII алфавита

commit 412fa02a829c2a6f2909d80318f5f69cb3fb6934
Author: PoTtaTto <MrPotatsus@yandex.ru>
Date: Mon Sep 11 16:52:30 2023 +0300

    Добавление функции вывода случайного числа
```

Рисунок 6 – История коммитов (1)

```

commit e9ae6318de9b863d49304ddf69a730bfe9e77770
Author: PoTtaTto <MrPotatsus@yandex.ru>
Date:   Mon Sep 11 16:47:46 2023 +0300

    Добавление функции таймера

commit 5e6661e2647ff1c03aa32cbcc542bef80f9341a0
Author: PoTtaTto <MrPotatsus@yandex.ru>
Date:   Mon Sep 11 16:39:12 2023 +0300

    Добавление интерпретатора и файла README.md

commit 0e5938ca1ddc718c8238b2a98c5c8bfbdd05f33f (origin/main, origin/HEAD)
Author: MrPotatsus <MrPotatsus@yandex.ru>
Date:   Mon Sep 11 15:52:07 2023 +0300

    Initial commit

```

Рисунок 7 – История коммитов (2)

Листинг полученной программы:

```

def set_timer(seconds: int):
    from time import sleep

    print('Timer started')
    for i in range(seconds):
        sleep(1)
        print(f'Wait time - {i + 1} sec.')
    print('Timer stopped')

def print_random_number(start: int, end: int):
    from random import randint
    print(randint(start, end), end='')

def print_random_letter():
    import string
    import random
    print(random.choice(string.ascii_letters), end='')

def print_password(length: int):
    set_timer(seconds=5)
    for i in range(length):
        if i % 2 == 0:
            print_random_letter()
        else:
            print_random_number(start=i, end=length)

if __name__ == '__main__':
    print_password(length=25)

```

```
Timer started
Wait time - 1 sec.
Wait time - 2 sec.
Wait time - 3 sec.
Wait time - 4 sec.
Wait time - 5 sec.
Timer stopped
i24a5k14N15S19r22f19p15x20M21H24D23m
Process finished with exit code 0
```

Рисунок 8 – Пример работы программы

6. Добавим описание в файл README.md:

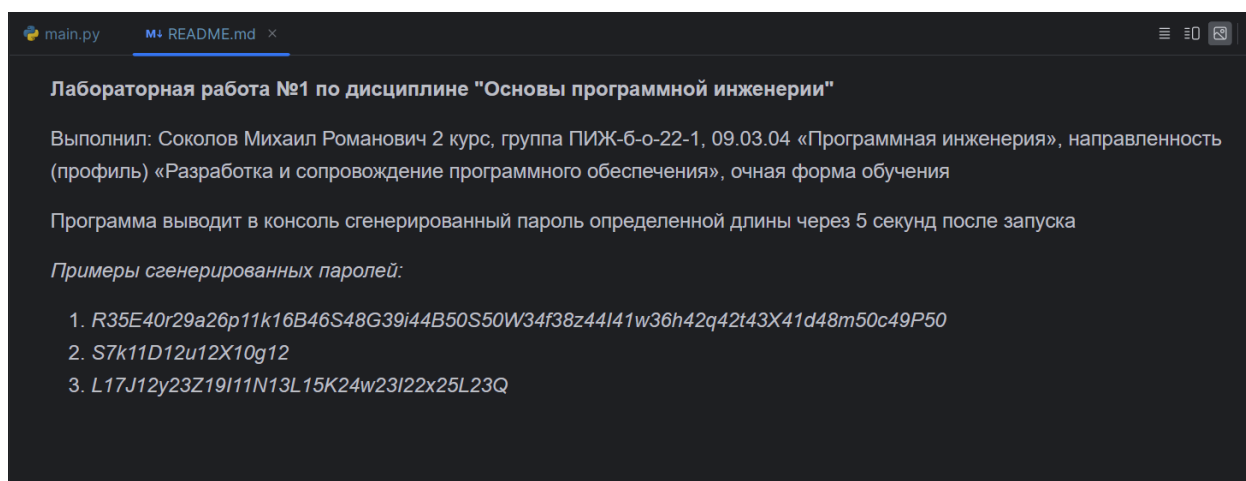


Рисунок 9 – Итоговый README.md файл

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?

У локальных самый большой недостаток является отсутствие возможности удаленной работы над проектом другими людьми. У ЦСКВ единая точка отказа, представленная центральным сервером. Если этот сервер

выйдет из строя на неопределенное время, 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 является распределенной СКВ (РСКВ).

4. В чем концептуальное отличие Git от других СКВ?

Подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.

5. Как обеспечивается целостность хранимых данных в Git?

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. Механизм, которым пользуется Git при вычислении хеш-сумм, называется SHA-1 хеш.

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

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

Зафиксированный значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

7. Что такое профиль пользователя в GitHub?

Профиль — это ваша публичная страница на GitHub, как и в социальных сетях. Когда вы ищете работу в качестве программиста, работодатели могут посмотреть ваш профиль GitHub и принять его во внимание, когда будут решать, брать вас на работу или нет. GitHub — это платформа для размещения

кода. Иными словами, это место, где разработчики могут хранить свои проекты и работать вместе.

8. Какие бывают репозитории в GitHub?

Репозитории могут быть открытыми (public) или закрытыми (private).

9. Укажите основные этапы модели работы с GitHub.

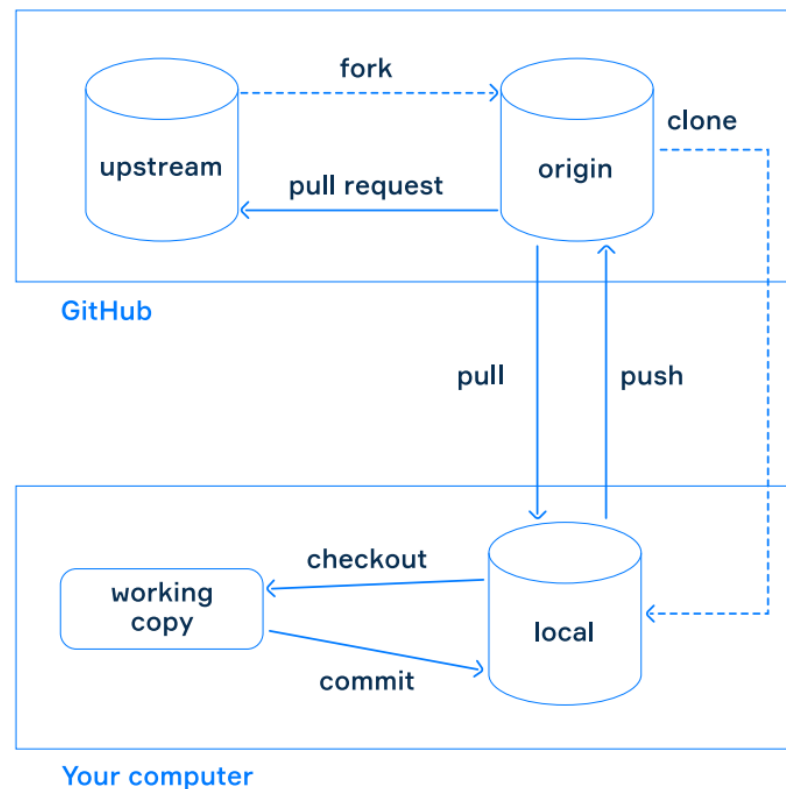


Рисунок 10 – Схема модели работы с GitHub

10. Как осуществляется первоначальная настройка Git после установки?

Чтобы убедиться, что git установлен используется команда «git version». После используются команды для привязки git к аккаунту GitHub: «git config --global user.name <YOUR_NAME>» и «git config --global user.email <EMAIL>».

11. Опишите этапы создания репозитория в GitHub.

Сначала обязательно вводится имя репозитория. Добавляется описание, но его можно оставить пустым. Выбирается вид доступа к репозиторию (public/private), а также файлы .gitignore и LICENSE.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Поддерживаемые лицензии: 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.

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

Клонирование репозитория на локальный компьютер осуществляется командой: «git clone <repository url>». Этот процесс передает данные с удаленного репозитория GitHub на локальный компьютер.

14. Как проверить состояние локального репозитория Git?

Можно воспользоваться 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?

Если использовать проверку статуса репозитория после добавления/изменения файлов в локальном репозитории, то файлы помечаются статусом modified. При использовании команды «git add» файлы, которые находились под статусом modified, получают статус staged и при дальнейшем использовании команды «git commit» фиксируются, создавая новый снимок. Команда «git push» передает изменения ветки на удаленный репозиторий GitHub.

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`.

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

Наиболее популярные сервисы, работающие с Git: BitBucket и GitLab.

Небольшое сравнение GitHub и Bitbucket:

GitHub:

- Владелец является компания Microsoft.
- Поддерживает только РСКВ Git.
- Огромное число сторонников и проектов.
- Обширная система интеграций со сторонними продуктами.

Bitbucket:

- Владелец является компания Atlassian.
- Поддерживает РСКВ Git и Mercurial.
- Интеграция только с другими продуктами Atlassian.

Решение выбора конкретной платформы зависит от проекта и предпочтений команды или автора.

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.

Примером программ с графическим интерфейсом для работы с Git могут являться: GitHub Desktop, Source Tree, GitKraken и т.д.

Рассмотрим GitHub Desktop:

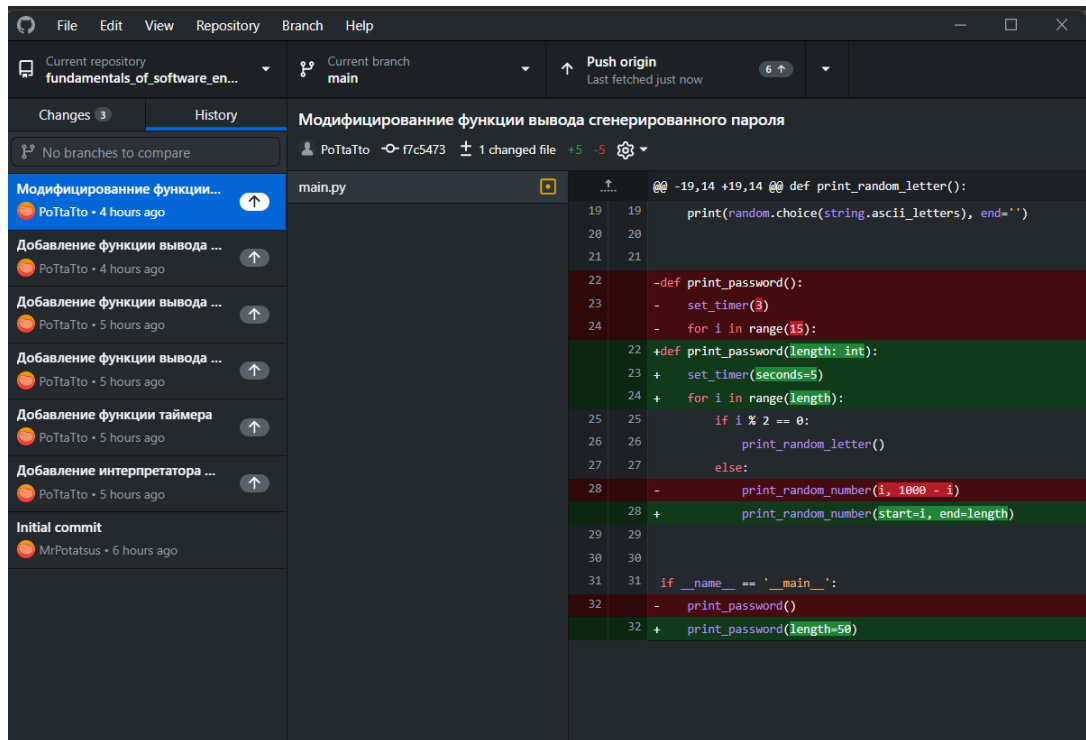


Рисунок 11 – Просмотр полной истории изменения проекта

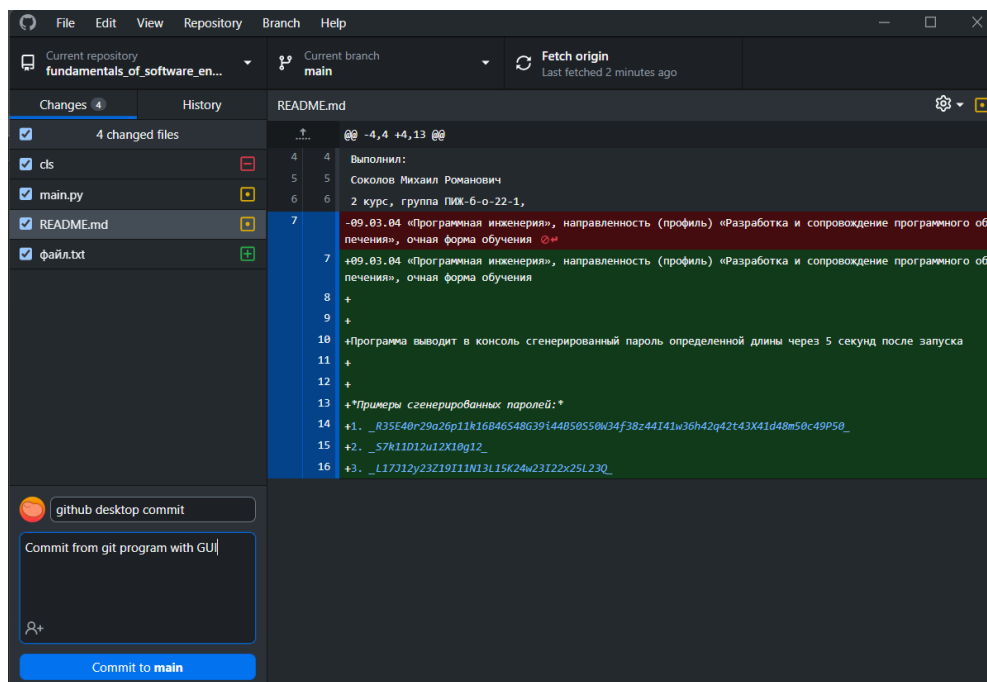


Рисунок 12 – Можно наблюдать статус файлов и их изменения. Совершение коммита происходит по нажатию кнопки в левом нижнем углу окна.

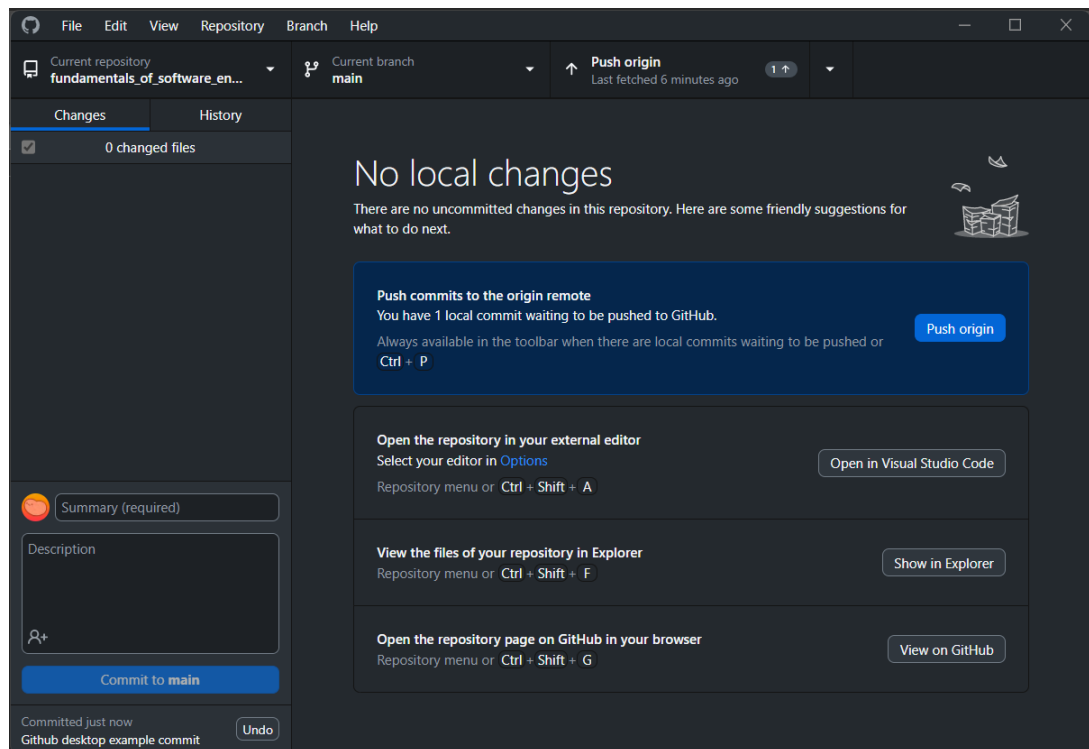


Рисунок 13 – После совершения коммита GitHub Desktop предлагает передать изменения в удаленный репозиторий GitHub