

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

Кафедра инфокоммуникаций

**«Исследование основных возможностей Git и GitHub»
Отчет по лабораторной работе № 1.1
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-22-1
Юрьев Илья Евгеньевич.
Подпись студента _____
Работа защищена « » _____ 20__ г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

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

Ход работы:

1. Создадим общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный язык программирования:

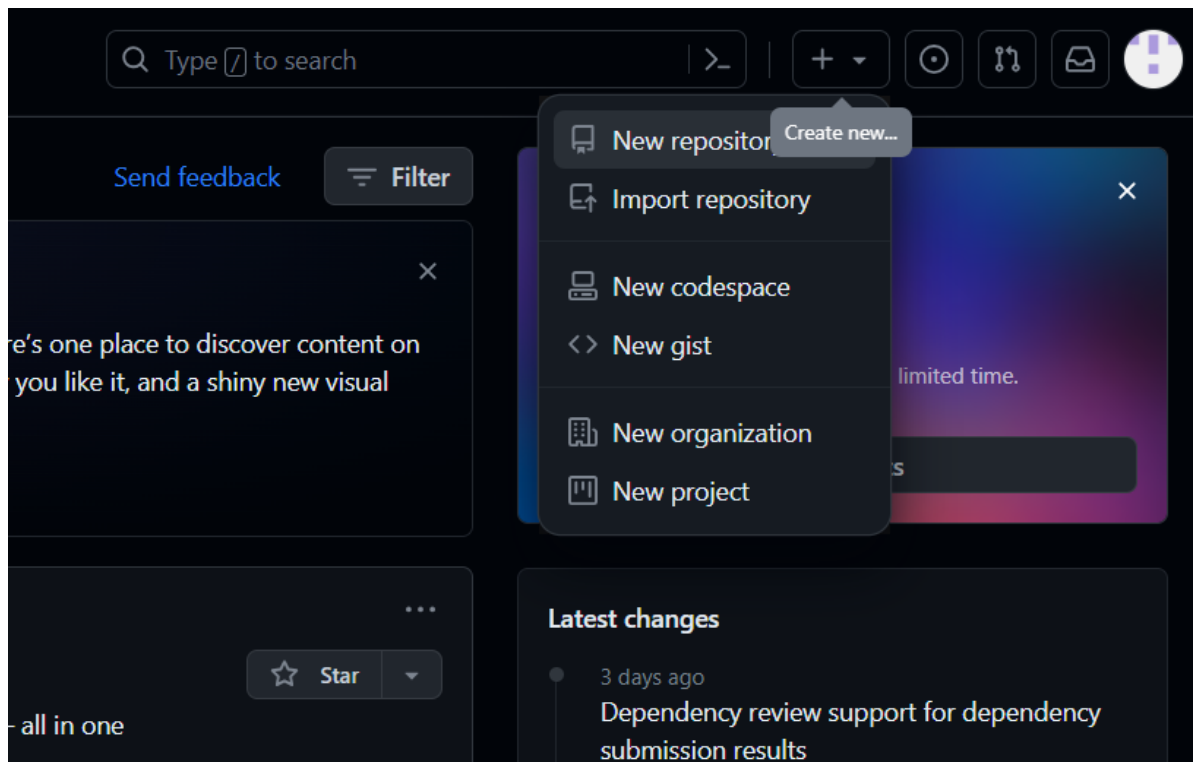



Рисунок 1 – Создание нового репозитория

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 * **Repository name ***

 daxstrong /

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

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:

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

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

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

[Create repository](#)

Рисунок 2 – Страница создания нового репозитория

.gitignore template

- Processing
- PureScript
- Python**
- Qooxdoo
- Qt
- R
- ROS
- Racket
- Rails
- Raku
- RhodesRhomobile

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Рисунок 3 – Выбор языка программирования

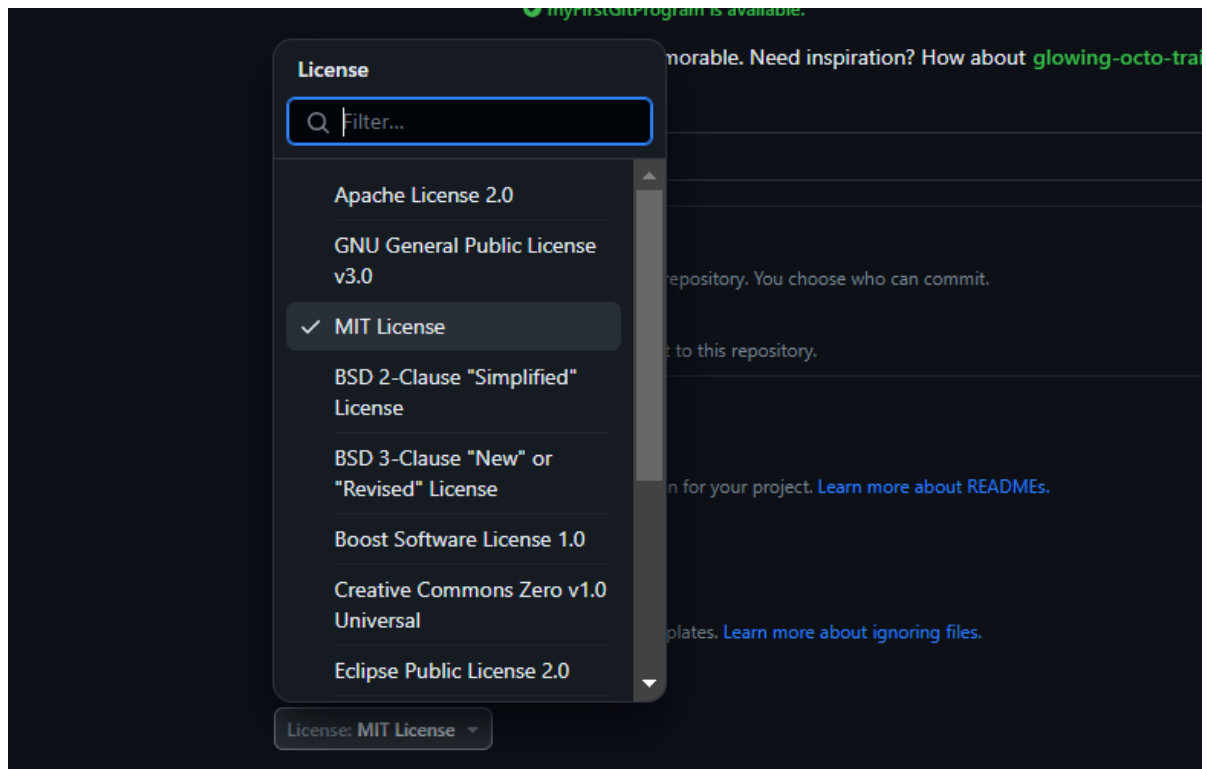


Рисунок 4 – Выбор MIT лицензии

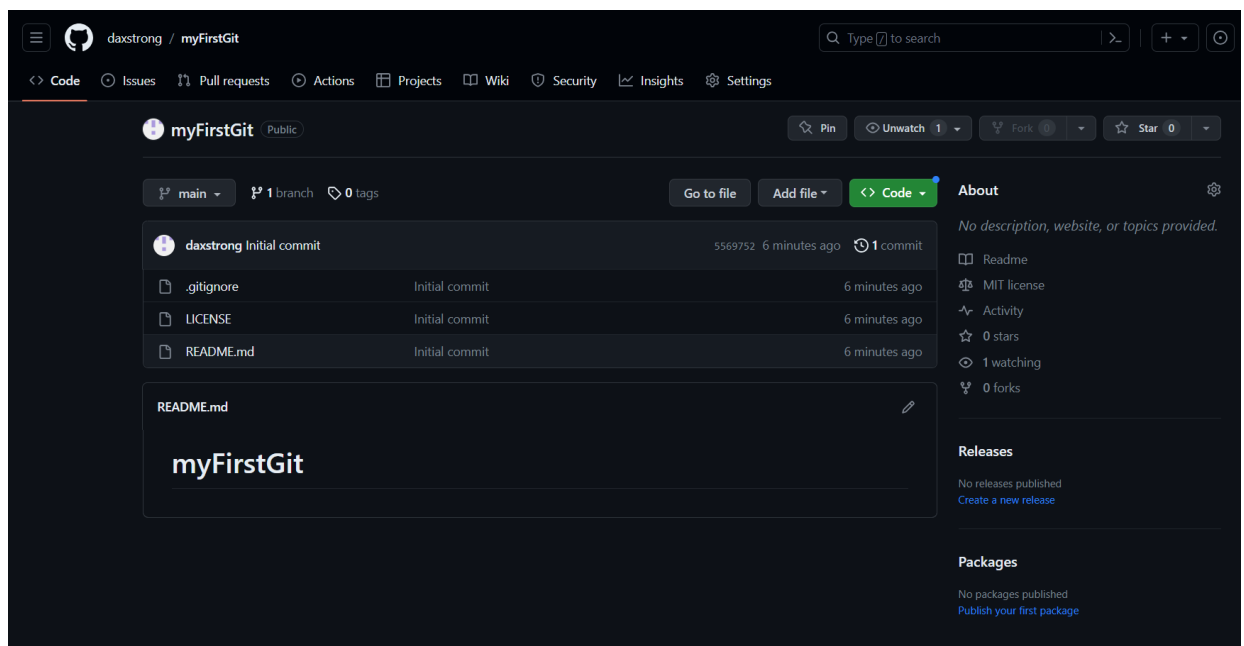
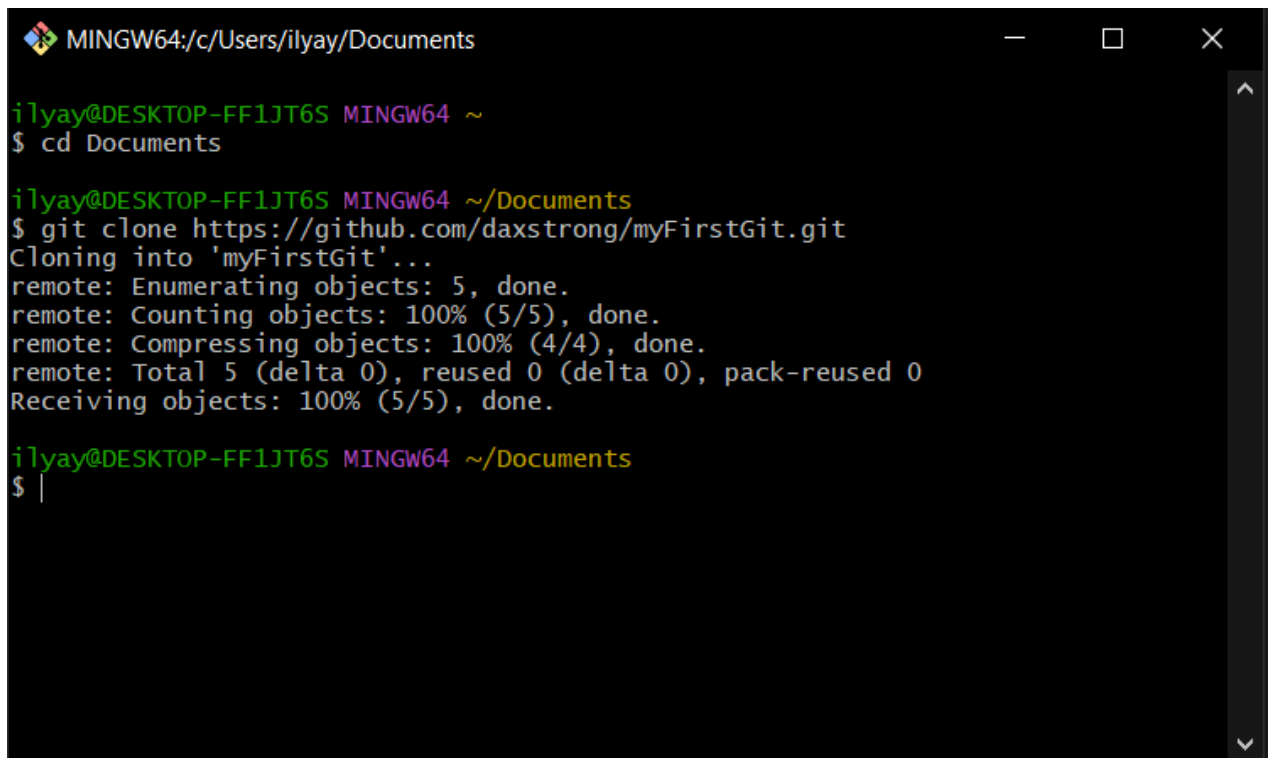


Рисунок 5 – Созданный репозиторий

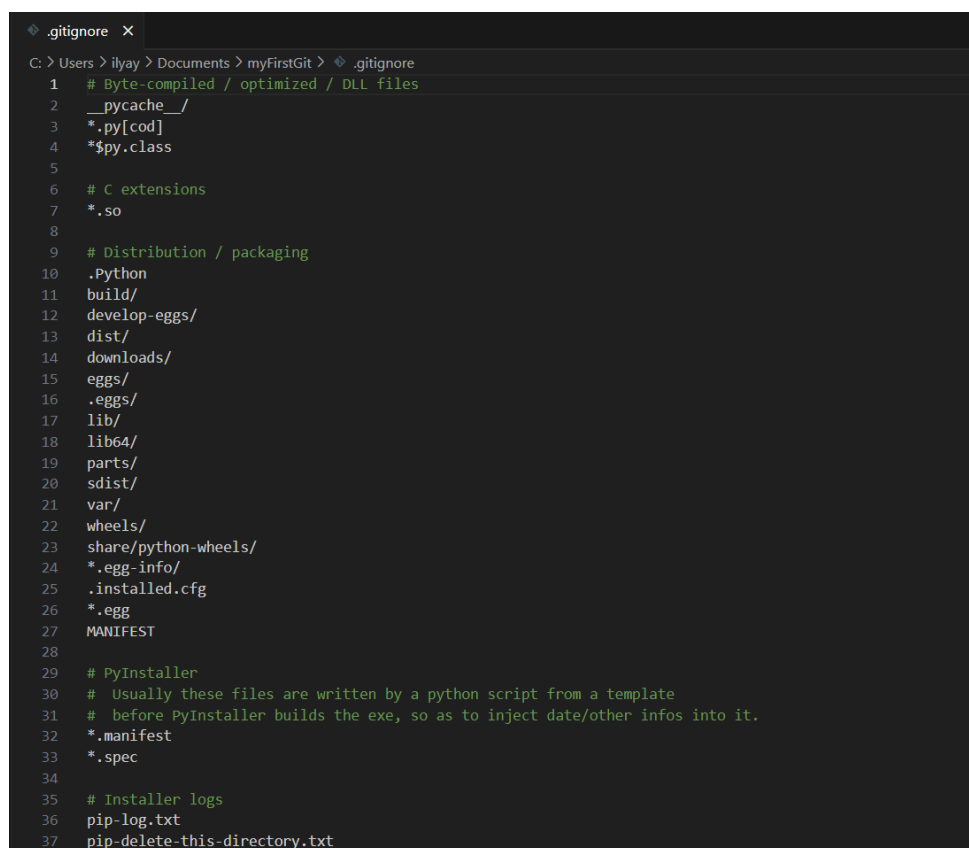
2. Выполним клонирование созданного репозитория на рабочий компьютер:



```
MINGW64:/c/Users/ilyay/Documents
ilyay@DESKTOP-FF1JT6S MINGW64 ~
$ cd Documents
ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents
$ git clone https://github.com/daxstrong/myFirstGit.git
Cloning into 'myFirstGit'...
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.
ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents
$ |
```

Рисунок 6 – Клонирование репозитория с помощью команды «git clone»

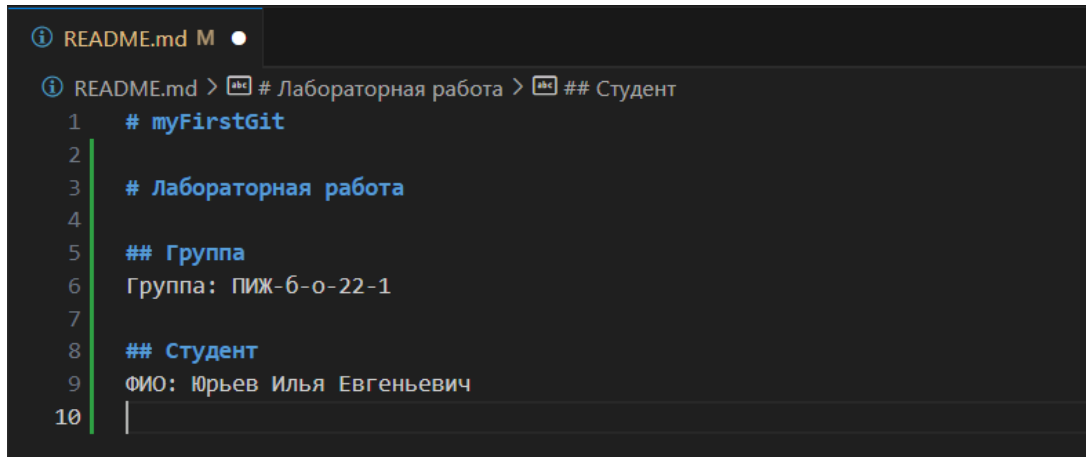
3. Дополним файл .gitignore необходимыми правилами для языка программирования Python:



```
.gitignore X
C:\> Users > ilyay > Documents > myFirstGit > .gitignore
1 # Byte-compiled / optimized / DLL files
2 __pycache__/
3 *.py[co]
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
```

Рисунок 7 – Содержимое файла «.gitignore»

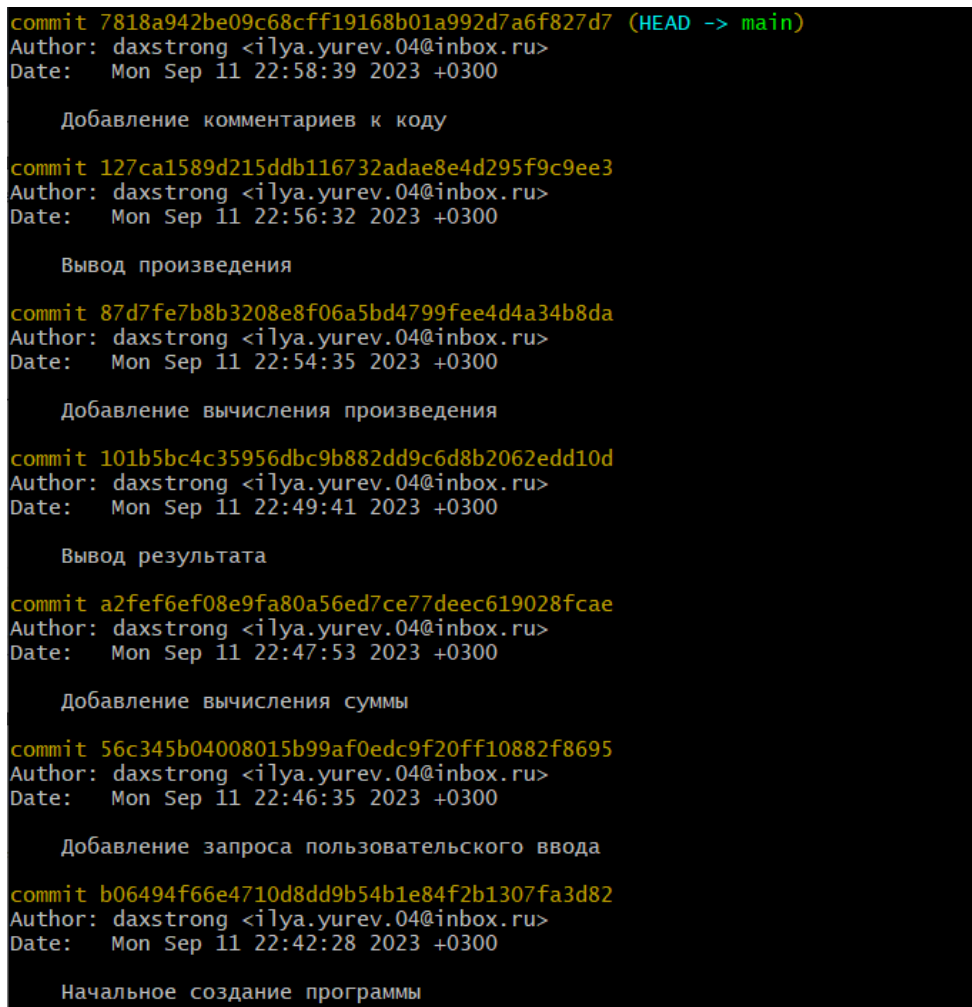
4. Добавим в файл README.md информацию о группе и ФИО студента, выполняющего лабораторную работу:



```
1 # myFirstGit
2
3 # Лабораторная работа
4
5 ## Группа
6 Группа: ПИЖ-6-о-22-1
7
8 ## Студент
9 ФИО: Юрьев Илья Евгеньевич
10 |
```

Рисунок 8 – Добавление информации в файл README.md

5. Напишем небольшую программу на выбранном языке программирования. Зафиксируем изменения в локальном репозитории:



```
commit 7818a942be09c68cff19168b01a992d7a6f827d7 (HEAD -> main)
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 11 22:58:39 2023 +0300

    Добавление комментариев к коду

commit 127ca1589d215ddb116732adae8e4d295f9c9ee3
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 11 22:56:32 2023 +0300

    Вывод произведения

commit 87d7fe7b8b3208e8f06a5bd4799fee4d4a34b8da
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 11 22:54:35 2023 +0300

    Добавление вычисления произведения

commit 101b5bc4c35956dbc9b882dd9c6d8b2062edd10d
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 11 22:49:41 2023 +0300

    Вывод результата

commit a2fef6ef08e9fa80a56ed7ce77deec619028fcae
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 11 22:47:53 2023 +0300

    Добавление вычисления суммы

commit 56c345b04008015b99af0edc9f20ff10882f8695
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 11 22:46:35 2023 +0300

    Добавление запроса пользовательского ввода

commit b06494f66e4710d8dd9b54b1e84f2b1307fa3d82
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 11 22:42:28 2023 +0300

    Начальное создание программы
```

Рисунок 9 – История коммитов

```
my_program.py
C: > Users > ilyay > Documents > myFirstGit > my_program.py > ...
1  # Программа для сложения двух чисел
2
3  # Получаем ввод от пользователя
4  num1 = float(input("Введите первое число: "))
5  num2 = float(input("Введите второе число: "))
6
7  # Вычисляем сумму
8  result1 = num1 + num2
9  # Вычисляем произведение
10 result2 = num1 * num2
11
12 # Выводим результат
13 print("Сумма чисел:", result1)
14 print("Произведение чисел:", result2)
15
```

Рисунок 10 – Код программы

6. Зафиксируем сделанные изменения в файле README.md:

```
C: > Users > ilyay > Documents > myFirstGit > README.md > # myFirstGit
1  # myFirstGit
2
3  # Лабораторная работа
4
5  ## Группа
6  Группа: ПИЖ-6-о-22-1
7
8  ## Студент
9  ФИО: Юрьев Илья Евгеньевич
10
11 ## Описание
12 Цель данной лабораторной работы исследовать базовые возможности системы контроля версий git и веб-сервиса для хостинга IT-проектов GitHub.
13
14 Программа просит пользователя ввести два числа и выводит сумму и произведение этих чисел.
```

Рисунок 11 – Итоговый файл README.md

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?

Многие люди в качестве метода контроля версий применяют копирование файлов в отдельную директорию (возможно даже, директорию с отметкой по времени, если они достаточно сообразительны). Данный подход очень распространён из-за его простоты, однако он невероятно сильно подвержен появлению ошибок. Можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели.

Самый очевидный минус — это единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, 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. Если жёсткий диск, на котором хранится центральная БД, повреждён, а своевременные бэкапы отсутствуют, вы потеряете всё — всю историю проекта, не считая единичных снимков репозитория, которые сохранились на локальных машинах разработчиков. Локальные СКВ страдают от той же самой проблемы: когда вся история проекта хранится в одном месте, вы рискуете потерять всё.

3. К какой СКВ относится Git?

Git является распределённой СКВ.

4. В чем концептуальное отличие Git от других СКВ?

Основное отличие Git от любой другой СКВ — это подход к работе со своими данными. Концептуально, большинство других систем хранят информацию в виде списка изменений в файлах. Эти системы представляют

хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени (обычно это называют контролем версий, основанным на различиях). Git не хранит и не обрабатывает данные таким способом. Вместо этого, 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. 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. Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его философии. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git.

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).

Если определённая версия файла есть в Git-директории, эта версия считается зафиксированной.

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. 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.

8. Какие бывают репозитории в GitHub?

Локальный — расположен на одном компьютере, и работать с ним может только один человек.

Централизованный — расположен на сервере, куда имеют доступ сразу несколько программистов.

Распределенный — самый удобный вариант с облачным хранилищем.

9. Укажите основные этапы модели работы с GitHub.

Сначала рассмотрим область GitHub. В нем есть два хранилища: upstream - это оригинальный репозиторий проекта, который вы скопировали, origin - ваш fork (копия) на GitHub, к которому у вас есть полный доступ. Чтобы перенести изменения с вашей копии в исходному репозиторий проекта, вам нужно сделать запрос на извлечение. Если вы хотите внести небольшие изменения в свою копию (fork), вы можете использовать вебинтерфейс GitHub. Однако такой подход не удобен при разработке программ, поскольку вам часто приходится запускать и отлаживать их локально. Стандартный способ - создать локальный клон удаленного репозитория и работать с ним локально, периодически внося изменения в удаленный репозиторий.

10. Как осуществляется первоначальная настройка Git после установки?

Чтобы убедиться, что Git был успешно установлен, введите команду ниже в терминале, чтобы отобразить текущую версию вашего Git:

```
git version
```

Если она сработала, давайте добавим в настройки Git ваше имя, фамилию и адрес электронной почты, связанный с вашей учетной записью GitHub:

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

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

11. Опишите этапы создания репозитория в GitHub.

В правом верхнем углу, рядом с аватаром есть кнопка с плюсиком, нажимая которую мы переходим к созданию нового репозитория.

В результате будет выполнен переход на страницу создания репозитория. Наиболее важными на ней являются следующие поля:

- Имя репозитория. Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиториях, которые вы создавали.
- Описание (Description). Можно оставить пустым.
- Public/private. Выбираем открытый (Public), НЕ ставим галочку “Initialize this repository with a README” (В README потом будет лежать какая-то основная информация, что же такое ваш проект и как с ним работать).
- .gitignore и LICENSE можно сейчас не выбирать. После заполнения этих полей нажимаем кнопку Create repository.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

GitHub поддерживает разнообразные типы лицензий, включая MIT License, GNU GPL, Apache License, Creative Commons Licenses, Unlicense и другие.

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

После создания репозитория его необходимо клонировать на ваш компьютер. Для этого на странице репозитория необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования.

Откройте командную строку или терминал и перейдите в каталог, куда вы хотите скопировать хранилище. Затем напишите `git clone` и введите адрес:

```
git clone https://github.com/kushedow/flask-html
```

14. Как проверить состояние локального репозитория Git?

Локальный репозиторий включает в себя все те же файлы, ветки и историю коммитов, как и удаленный репозиторий. Введите эту команду, чтобы проверить состояние вашего репозитория:

```
git status
```

15. Как изменяется состояние локального репозитория Git после выполнения следующих операций:

Команда `git add` добавляет изменение из рабочего каталога в раздел проиндексированных файлов. Она сообщает Git, что вы хотите включить изменения в конкретном файле в следующий коммит.

Команда `git commit` сохраняет снимок состояния из раздела проиндексированных файлов в истории коммитов репозитория.

Важно, что коммит добавляет изменения только в ваш локальный репозиторий. Если вы хотите распространить их в исходный репозиторий на GitHub, вам нужно использовать `push`. В первый раз вам также необходимо отправить свою локальную ветку, потому что она не существует в удаленном репозитории.

```
git push --set-upstream origin edit-readme
```

В следующий раз сделать это будет уже проще:

```
git push
```

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.

GitHub и GitLab - популярные сервисы для работы с Git. GitHub широко используется с большим сообществом, интеграцией инструментов и ограниченными бесплатными возможностями для частных репозиторий. GitLab предоставляет мощные инструменты CI/CD, самостоятельное развертывание и более функциональные бесплатные версии для частных репозиторий. Выбор зависит от потребностей и предпочтений.

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.

Некоторые GUI-средства для Git включают GitHub Desktop, GitKraken, SourceTree и TortoiseGit. С их помощью можно удобно выполнить операции Git, такие как клонирование репозитория, создание коммитов и отправка изменений на удаленный репозиторий.