**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРОТОРНОЙ РАБОТЕ № 2.**

*дисциплина: Архитектура компьютера гг*

Студент: Цоппа Е.Э.

Группа: НКАбд-06-23

**МОСКВА**

2023 г.

**Содержание**

1 Цель работы…………………………………………………………………………… 5

2 Задание………………………………………………………………………………… 6

3 Теоретическое введение……………………………………………………………… 7

4 Выполнение лабораторной работы …………………………………………………..9

4.1 Настройка GitHub…………………………..…………………………………. 9

4.2 Базовая настройка Git ……………………………………………………..…. 9

4.3 Создание SSH-ключа ………………………………………………………...10

4.4 Создание рабочего пространства и репозитория курса на основе шаблона…………………………………………………………………….…13

4.5 Создание репозитория курса на основе шаблона ………………………… 13

4.6 Настройка каталога курса…………………………………………………….16

4.7 Выполнение заданий для самостоятельной работы ……………………….18

5 Выводы……………………………………………………………………………… 22

6 Список литературы ………………………………………………………………….23

**Список иллюстраций**

4.1 Заполнение данных учетной записи GitHub ………………………………………9

4.2 Аккаунт GitHub …………………………………………………………………..….9

4.3 Предварительная конфигурация git ………………………………………………10

4.4 Настройка кодировки ……………………………………………………………...10

4.5 Создание имени для начальной ветки …………………………………………....10

4.6 Параметр autocrlf ……………………………………………………………….….10

4.7 Параметр safecrlf …………………………………………………………………...10

4.8 Генерация SSH-ключа ……………………………………………………………..11

4.9 Установка утилиты xclip…………………………………………………………...11

4.10 Копирование содержимого файла………………………………………………..12

4.11 Окно SSH and GPG keys ………………………………………………………….12

4.12 Добавление ключа ………………………………………………………………..13

4.13 Создание рабочего пространства……………………………………………...…13

4.14 Страница шаблона для репозитория ……………………………………………14

4.15 Окно создания репозитория ……………………………………………………..14

4.16 Созданный репозиторий …………………………………………………………15

4.17 Перемещение между директориями …………………………………………….15

4.18 Клонирование репозитория ……………………………………………………...15

4.19 Окно с ссылкой для копирования репозитория ………………………………...16

4.20 Перемещение между директориями ………………………………………….…16

4.21 Удаление файлов…………………………………………………………………. 16

4.22 Создание каталогов …………………………………………………………….…16

4.23 Добавление и сохранение изменений на сервере ………………………………17

4.24 Выгрузка изменений на сервер ………………………………………………..…17

4.25 Страница репозитория………………………………………………………….…18

4.26 Создание файла ...…………………………………………………………………18

4.27 Меню приложений ………………………………………………………………..18

4.28 Работа с отчетом в текстовом процессоре ………………………………………19

4.29 Перемещение между директориями……………………………………………..19

4.30 Проверка местонахождения файлов……………………………………………..19

4.31 Копирование файла ………………………………………………………………19

4.32 Добавление файла на сервер ………………………………………………….….20

4.33 Подкаталоги и файлы в репозитории ……………………………………………20

4.34 Отправка в центральный репозиторий сохраненных изменений………………20

4.35 Страница каталога в репозитории …………………………………………….…20

4.36 Страница последних изменений в репозитории……………………...…………21

4.37 Каталог lab01/report…………………………………………….…………………21

4.38 Каталог lab02/report ………………………………………………………………21

**1 Цель работы**

Целью данной работы является изучить идеологию и применение средств контроля версий, а также приобрести практические навыки по работе с системой git.

**2 Задание**

1. Настройка GitHub.

2. Базовая настройка Git.

3. Создание SSH-ключа.

4. Создание рабочего пространства и репозитория курса на основе шаблона.

5. Создание репозитория курса на основе шаблона.

6. Настройка каталога курса.

7. Выполнение заданий для самостоятельной работы.

**3 Теоретическое введение**

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить изменения, сделанные разными участниками, вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет 7 другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом привилегированный доступ только одному пользователю, работающему с файлом. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией. Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений). Затем можно вносить изменения в локальном дереве и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории.

**4 Выполнение лабораторной работы**

**4.1 Настройка GitHub**

Создаю учетную запись на сайте GitHub (рис. 4.1). Далее я заполнила основные данные учетной записи.

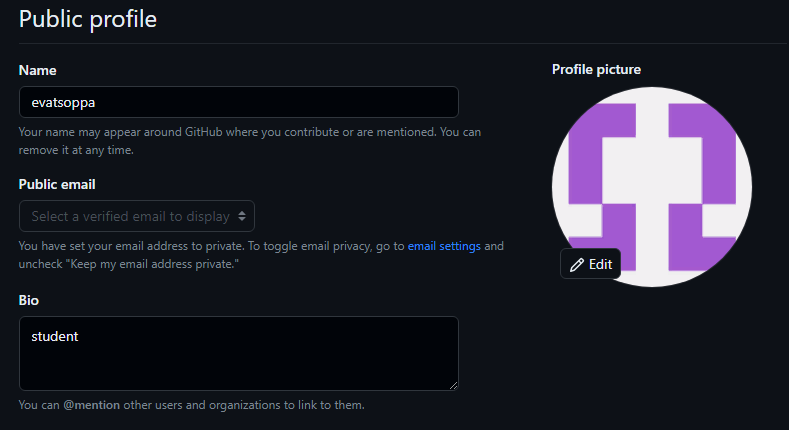


Рис. 4.1. Заполнение данных учетной записи GitHub

Аккаунт создан (рис. 4.2).

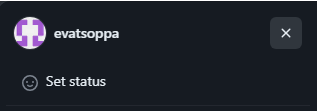


Рис. 4.2 Аккаунт GitHub

**4.2 Базовая настройка Git**

Открываю виртуальную машину, затем открываю терминал и делаю предварительную конфигурацию git. Ввожу команду git config –global user.name “”, указывая свое имя и команду git config –global user.email “work@mail”, указывая в ней электронную почту владельца, то есть мою (рис. 4.3).

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_105.png

Рис. 4.3. Предварительная конфигурация git

Настраиваю utf-8 в выводе сообщений git для корректного отображения символов (рис. 4.4).

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_106.png

Рис. 4.4. Настройка кодировки

Задаю имя «master» для начальной ветки (рис. 4.5).

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_107.png

Рис. 4.5. Создание имени для начальной ветки

Задаю параметр autocrlf со значением input, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах (рис. 4.6). CR и LF – это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах.

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_108.png

Рис. 4.6 Параметр autocrlf

Задаю параметр safecrlf со значением warn, так Git будет проверять преобразование на обратимость (рис. 4.7). При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_109.png

Рис. 4.7. Параметр safecrlf

**4.3 Создание SSH-ключа**

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый). Для этого ввожу команду ssh-keygen -C “Имя Фамилия, work@email”, указывая имя владельца и электронную почту владельца (рис. 4.8). Ключ автоматически сохранится в каталоге ~/.ssh/.

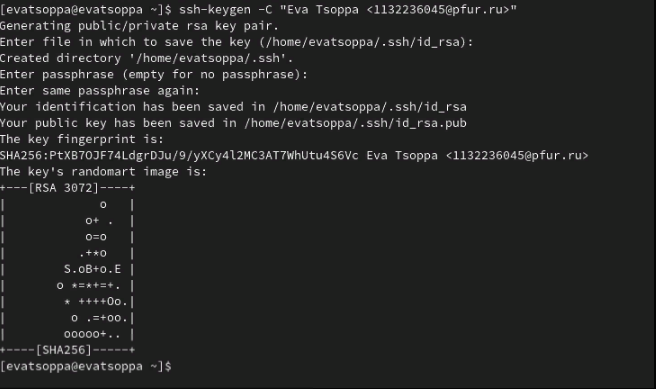


Рис. 4.8. Генерация SSH-ключа

Xclip – утилита, позволяющая скопировать любой текст через терминал. Сначала провожу ее установку. (рис. 4.9).

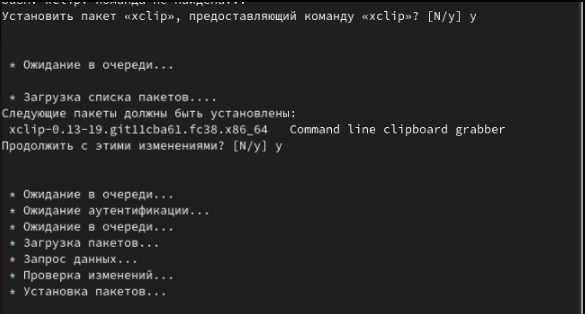


Рис. 4.9. Установка утилиты xclip

Копирую открытый ключ из директории, в которой он был сохранен, с помощью утилиты xclip (рис. 4.10).

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_112.png

Рис. 4.10. Копирование содержимого файла

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» (рис. 4.11).

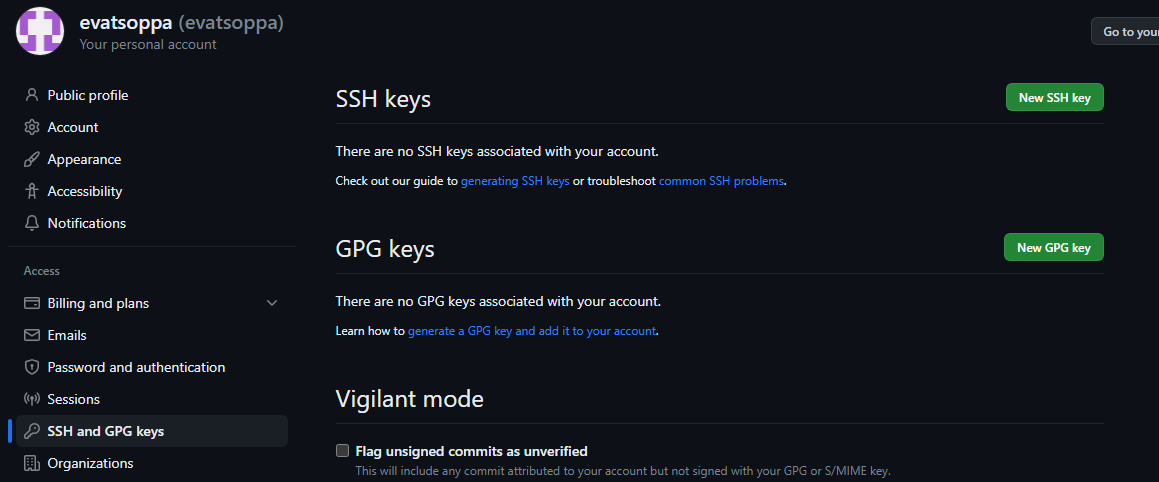


Рис. 4.11. Окно SSH and GPG keys

Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы завершить добавление ключа (рис. 4.12).

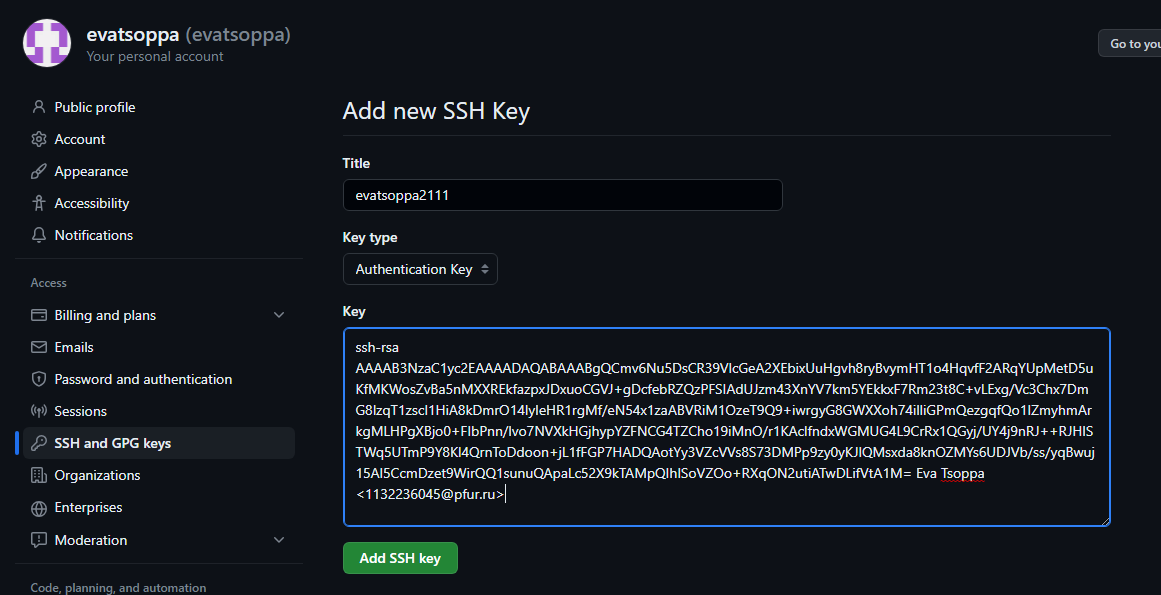


Рис. 4.12. Добавление ключа

**4.4 Создание рабочего пространства и репозитория курса на основе шаблона**

Закрываю браузер, открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты mkdir, блягодаря ключу -p создаю все директории после домашней ~/work/study/2023-2024/“Архитектура компьютера” рекурсивно. Далее проверяю с помощью ls, действительно ли были созданы необходимые мне каталоги (рис. 4.13).

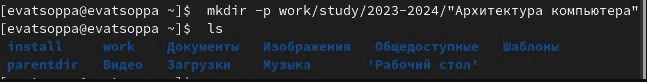


Рис. 4.13. Создание рабочего пространства

**4.5 Создание репозитория курса на основе шаблона**

В браузере перехожу на страницу репозитория с шаблоном курса по адресу https://github.com/yamadharma/course-directory-student-template. Далее выбираю «Use this template», чтобы использовать этот шаблон для своего репозитория (рис. 4.14).

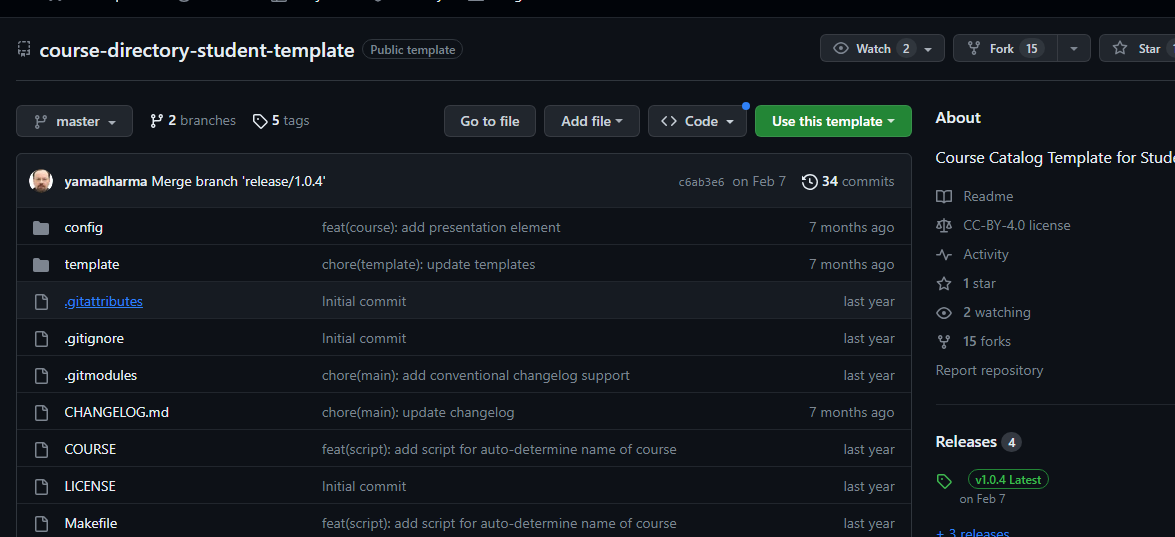


Рис. 4.14. Страница шаблона для репозитория

В открывшемся окне задаю имя репозитория (Repository name): study\_2023–2024\_arhpc и создаю репозиторий, нажимаю на кнопку «Create repository from template» (рис. 4.15).

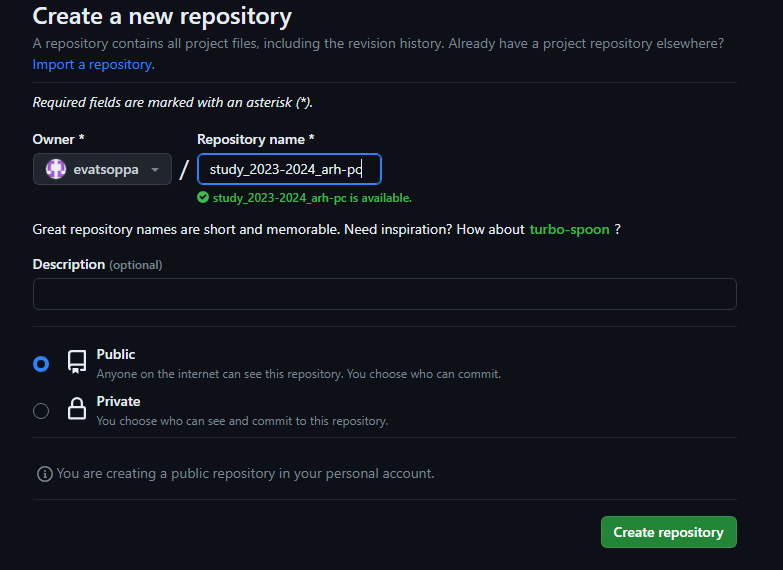


Рис. 4.15. Окно создания репозитория

Репозиторий создан (рис. 4.16).

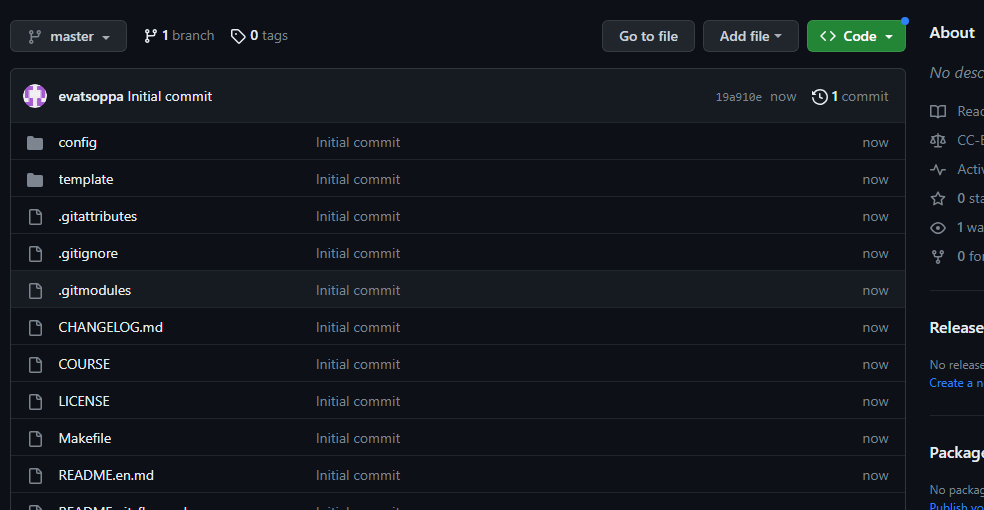


Рис. 4.16. Созданный репозиторий

Через терминал перехожу в созданный каталог курса с помощью утилиты cd (рис. 4.17).

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_118.png

Рис. 4.17. Перемещение между директориями

Клонирую созданный репозиторий с помощью команды git clone –recursive git@github.com:/study\_2023–2024\_arh-pc.git arch-pc (рис. 4.18).

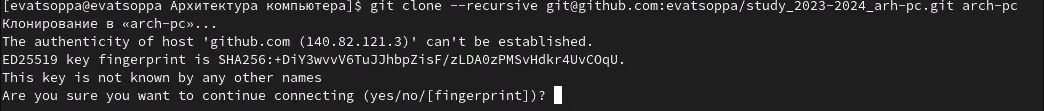


Рис. 4.18. Клонирование репозитория

Копирую ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. 4.19).

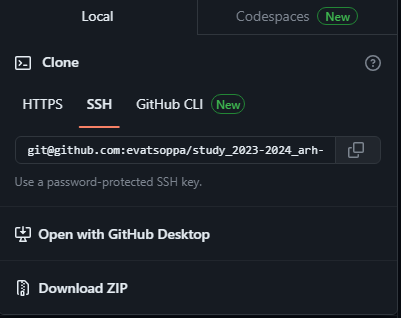


Рис. 4.19. Окно с ссылкой для копирования репозитория

**4.6 Настройка каталога курса**

Перехожу в каталог arch-pc с помощью утилиты cd (рис. 4.20).

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_152.png

Рис. 4.20. Перемещение между директориями

Удаляю лишние файлы с помощью утилиты rm (рис. 4.21).

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_153.png

Рис. 4.21. Удаление файлов

Создаю необходимые каталоги (рис. 4.22).

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_154.png

Рис. 4.22. Создание каталогов

Отправляю созданные каталоги с локального репозитория на сервер: добавляю все созданные каталоги с помощью git add, комментирую и сохраняю изменения на сервере как добавление курса с помощью git commit (рис. 4.23).

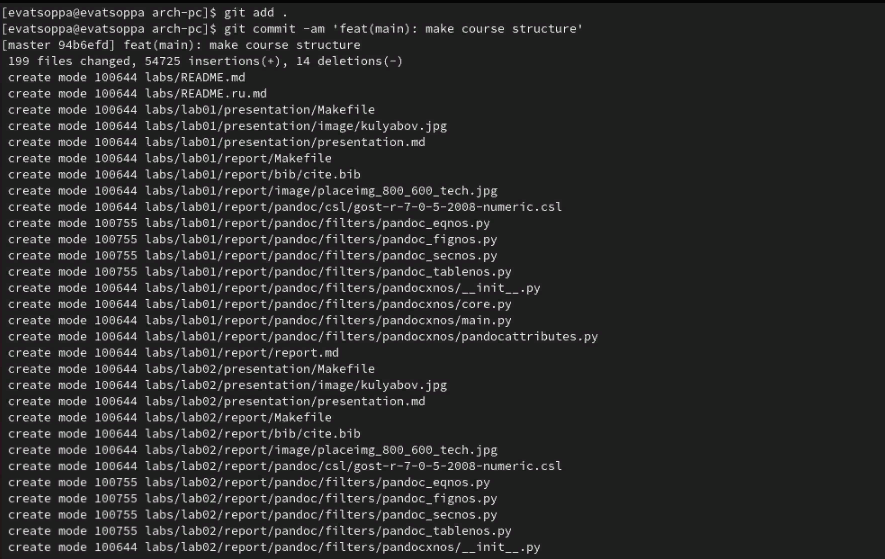


Рис. 4.23. Добавление и сохранение изменений на сервере

Отправляю все на сервер с помощью push (рис. 4.24).

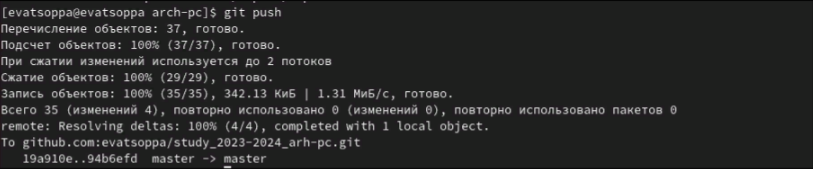


Рис. 4.24. Выгрузка изменений на сервер

Проверяю правильность выполнения работы сначала на самом сайте GitHub (рис. 4.25).

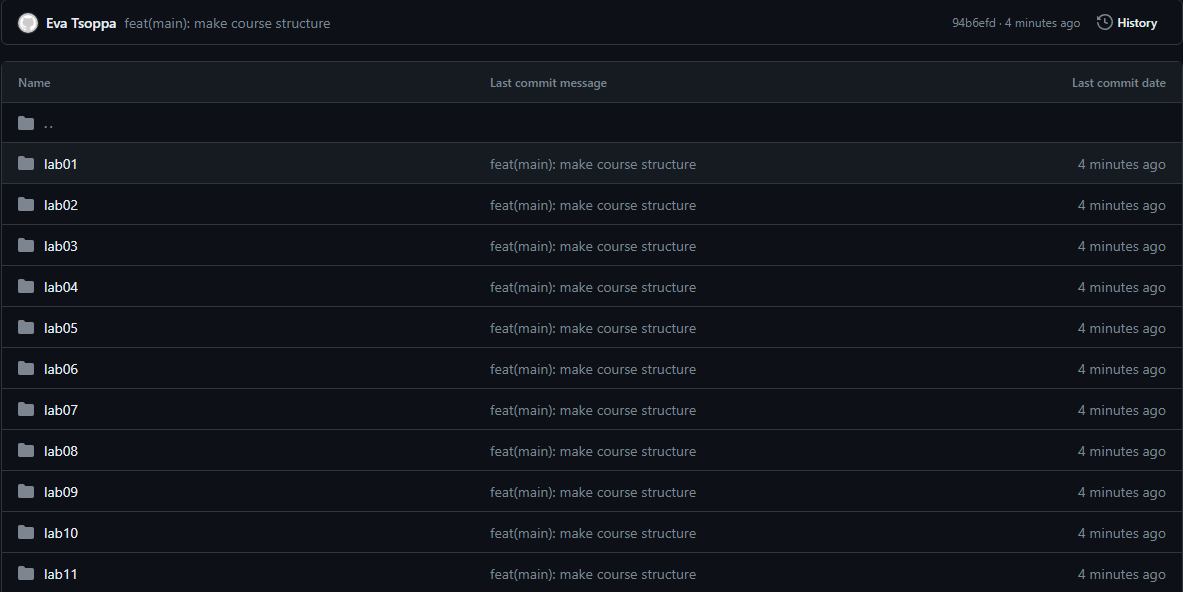


Рис. 4.25. Страница репозитория

**4.7 Выполнение заданий для самостоятельной работы**

1. Перехожу в директорию labs/lab02/report с помощью утилиты cd. Создаю в каталоге файл для отчета по второй лабораторной работе с помощью утилиты touch (рис. 4.26).

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_157.png

Рис. 4.26. Создание файла

Оформить отчет я смогу в текстовом процессоре LibreOffice Writer, найдя его в меню приложений (рис. 4.27).

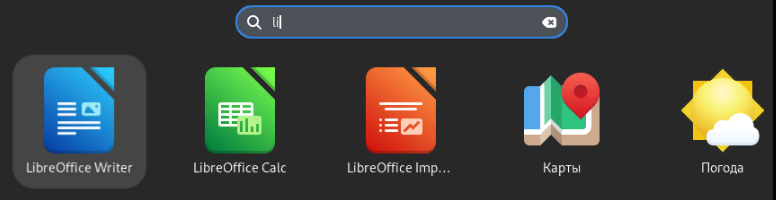


Рис. 4.27. Меню приложений

После открытия текстового процессора открываю в нем созданный файл и могу начать в нем работу над отчетом (рис. 4.28).

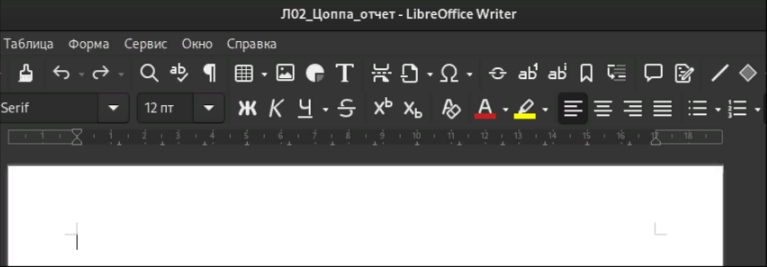


Рис. 4.28. Работа с отчетом в текстовом процессоре

2. Перехожу из подкаталога lab02/report в подкаталог lab01/report с помощью утилиты cd (рис. 4.29).

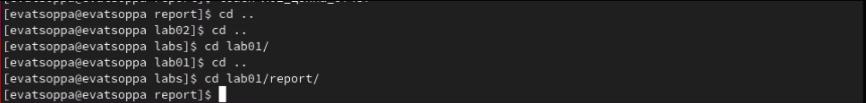


Рис.4.29. Перемещение между директориями

Проверяю местонахождение файлов с отчетами по первой и второй лабораторным работам. Они должны быть в подкаталоге домашней директории «Загрузки», для проверки использую команду ls (рис. 4.30).

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_161.png

Рис. 4.30. Проверка местонахождения файлов

Копирую первую лабораторную с помощью утилиты cp и проверяю правильность выполнения команды cp с помощью ls (рис. 4.31).

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_162.png

Рис. 4.31. Копирование файлов

3. Добавляю с помощью команды git add в коммит созданные файлы: Л01\_Цоппа\_ отчет (рис. 4.32).

C:\Users\User\Desktop\Алгбра дз\Скрины алг 1\Screenshot_163.png

Рис.4.32. Добавление файла на сервер

Сохраняю изменения на сервере командой git commit -m “…”, поясняя, что добавила файлы.

То же самое делаю для отчета по второй лабораторной работе: перехожу в директорию labs/lab02/report с помощью cd, добавляю с помощью git add нужный файл, сохраняю изменения с помощью git commit (рис. 4.33).

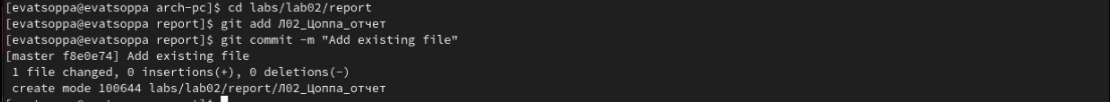


Рис.4.33. Подкаталоги и файлы в репозитории

Отправляю в центральный репозиторий сохраненные изменения командой git push -f origin master (рис. 4.34).

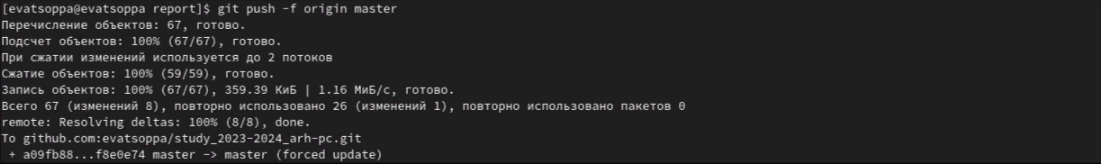


Рис.4.34. Отправка в центральный репозиторий сохраненных изменений

Проверяю на сайте GitHub правильность выполнения заданий. Вижу, что пояснение к совершенным действиям отображается (рис. 4.35).

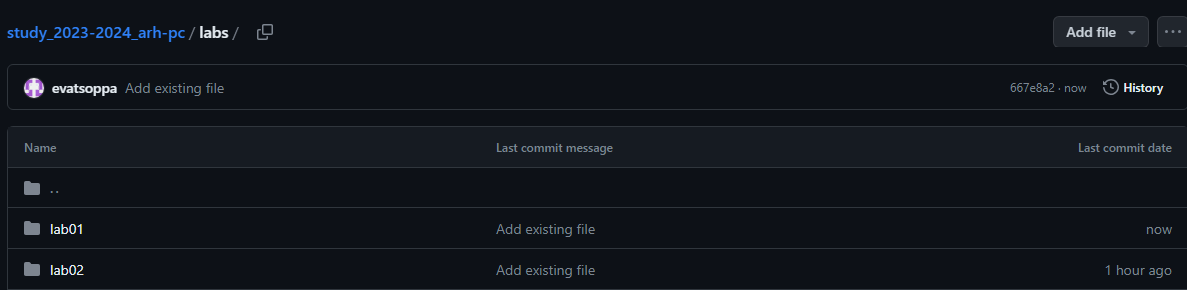


Рис.4.35. Страница каталога в репозитории

При просмотре изменений так же вижу, что был добавлен файл с отчетом по лабораторной работе (рис. 4.36).



Рис.4.36. Страница последних изменений в репозитории

Вижу, что отчеты по лабораторным работам находятся в соответствующих каталогах репозитория: отчет по первой - в lab01/report (рис. 4.37), по второй – в lab02/report (рис. 4.38).

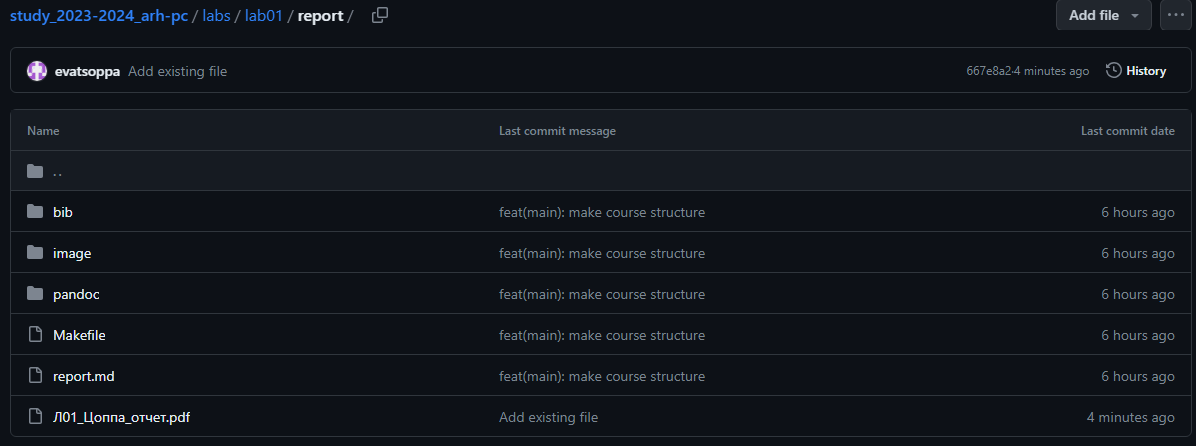


Рис.4.37. Каталог lab01/report

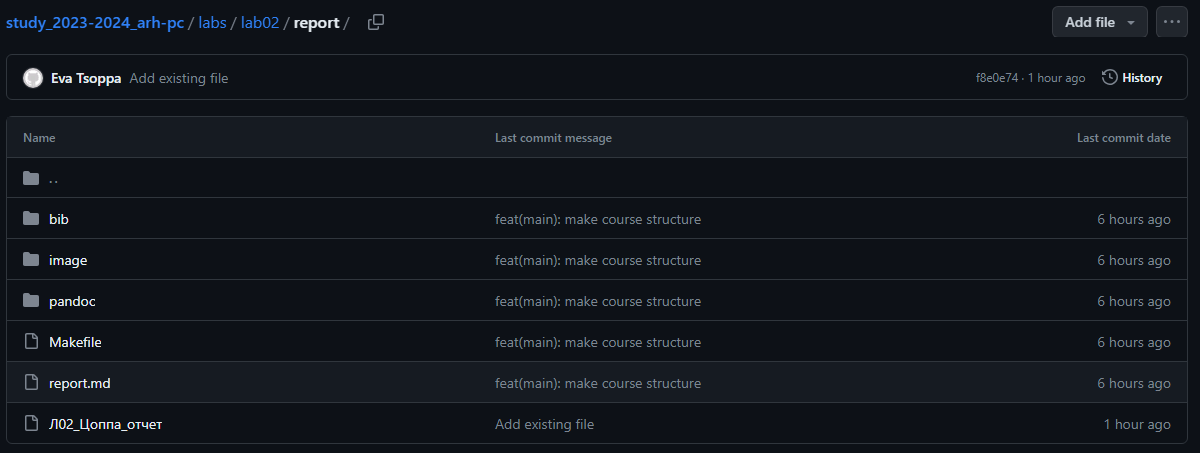


Рис.4.38. Каталог lab02/report

**5 Выводы**

При выполнении данной лабораторной работы я изучила идеологию и применение средств контроля версий, а также приобрела практические навыки по работе с системой git.

**6 Список литературы**

1. Архитектура ЭВМ

2. Git - gitattributes Документация