**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2**

*дисциплина: Архитектура компьютера*

Студент: Гомазкова Алина

Группа: НКАбд-02-23

**СОДЕРЖАНИЕ:**

**1) Цель работы ……………………………………………………………………………………………………………….**

**2) Теоретическое введение …………………………………………………………………………………………….**

**3) Настройка github ………………………………………………………………………………………………………..**

**4) Базовая настройка git …………………………………………………………………………………………………**

**5) Создание SSH ключа …………………………………………………………………………………………………..**

**6) Сознание рабочего пространства и репозитория курса на основе шаблона .…………..**

**7) Сознание репозитория курса на основе шаблона ……………………………………………………..**

**8) Настройка каталога курса …………………………………………………………………………………………..**

**9) Вывод ………………………………………………………………………………………………………………………….**

**10) Список литературы …………………………………………………………………………………………………..**

**1) Цель работы**

Целью работы является изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

**2) 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 (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет 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.

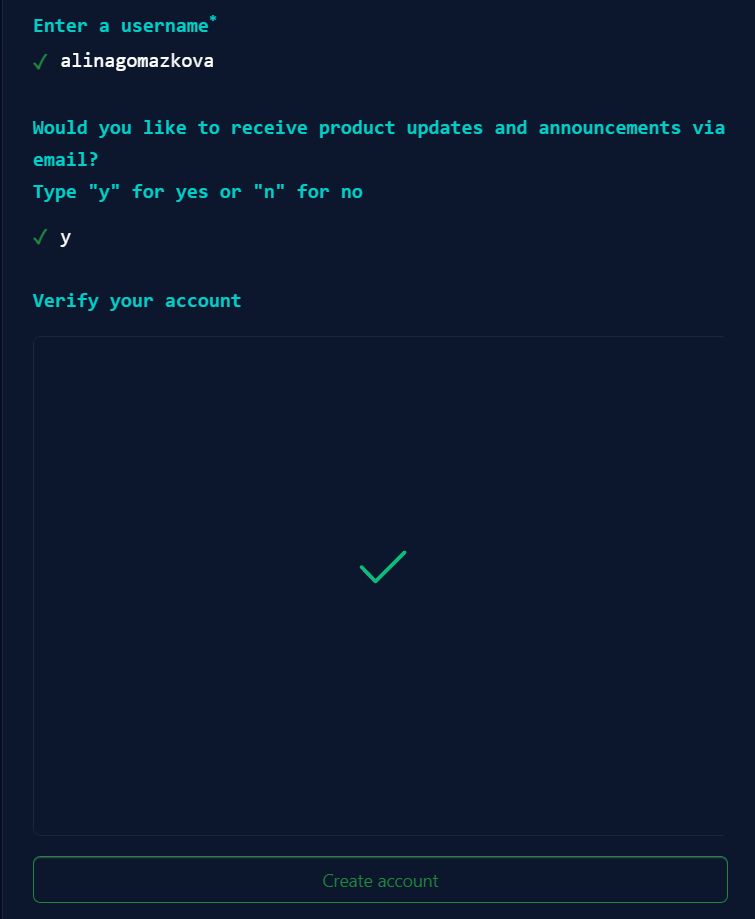
Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно 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.

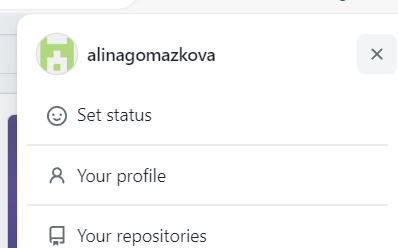
В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

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.

**3) Настройка github**

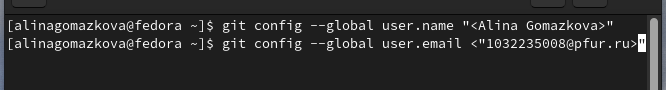
Создаю учётную запись на сайте https://github.com/ и заполняю основные данные.





**4) Базовая настройка git**

Сначала сделаю предварительную конфигурацию git. Открваю терминал и ввожу следующие команды git config --global user.name "<Name Surname>" git config --global user.email "<work@mail>" , указав имя и email владельца репозитория.



Настроим utf-8 в выводе сообщений git:



Задаю имя «master» для начальной ветки



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

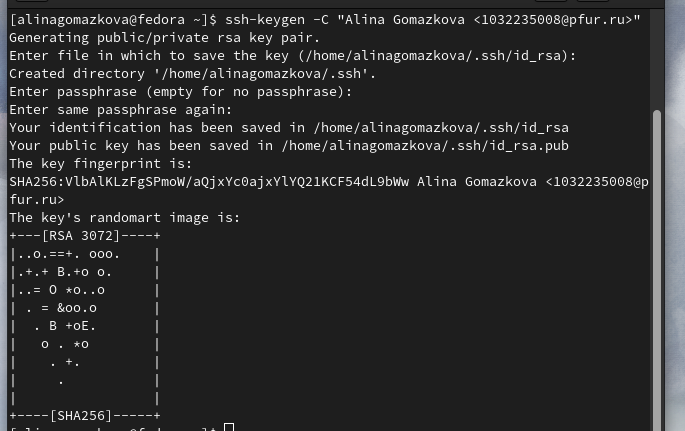


Задаю параметр safecrlf со значением warn, так Git будет проверять преобразование на обратимость. При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.

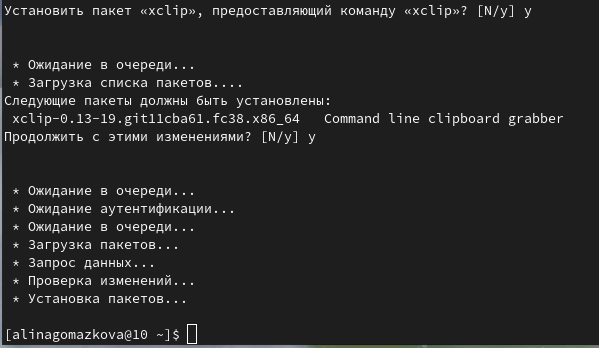


**5) Создание SSH ключа**

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](mailto:work@email)", указывая имя владельца и электронную почту владельца. Ключ автоматически сохранится в каталоге ~/.ssh/.

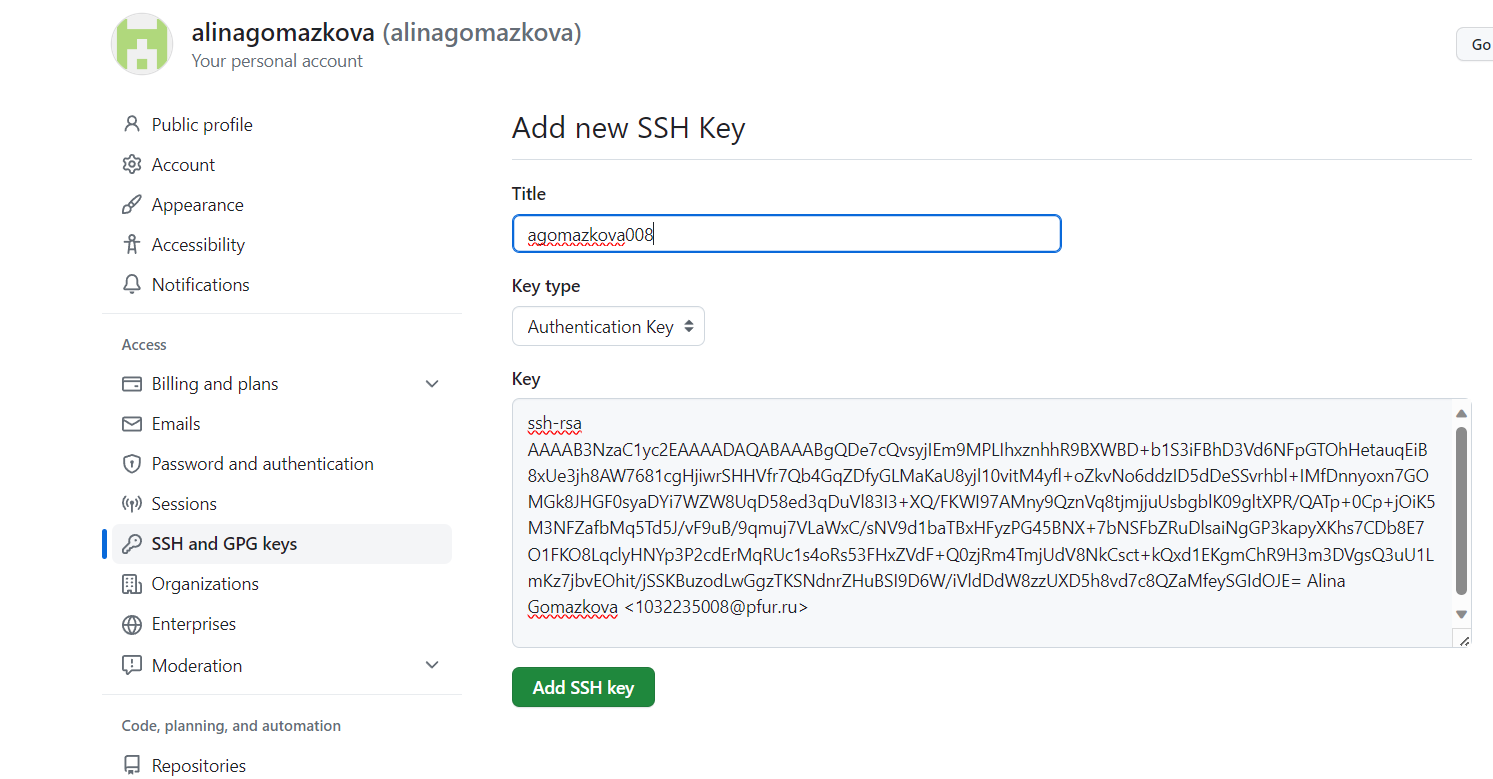


Копирую открытый ключ из директории, в которой он был сохранен, с помощью утилиты xclip



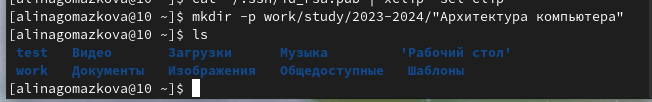


Далее необходимо загрузить сгенерённый открытый ключ. Для этого зайти на сайт http: //github.org/ под своей учётной записью и перейти в меню Setting . После этого выбрать в боковом меню SSH and GPG keys и нажать кнопку New SSH key . Скопировав из локальной консоли ключ в буфер обмена. Вставляем ключ в появившееся на сайте поле и указываем для ключа имя (Title).



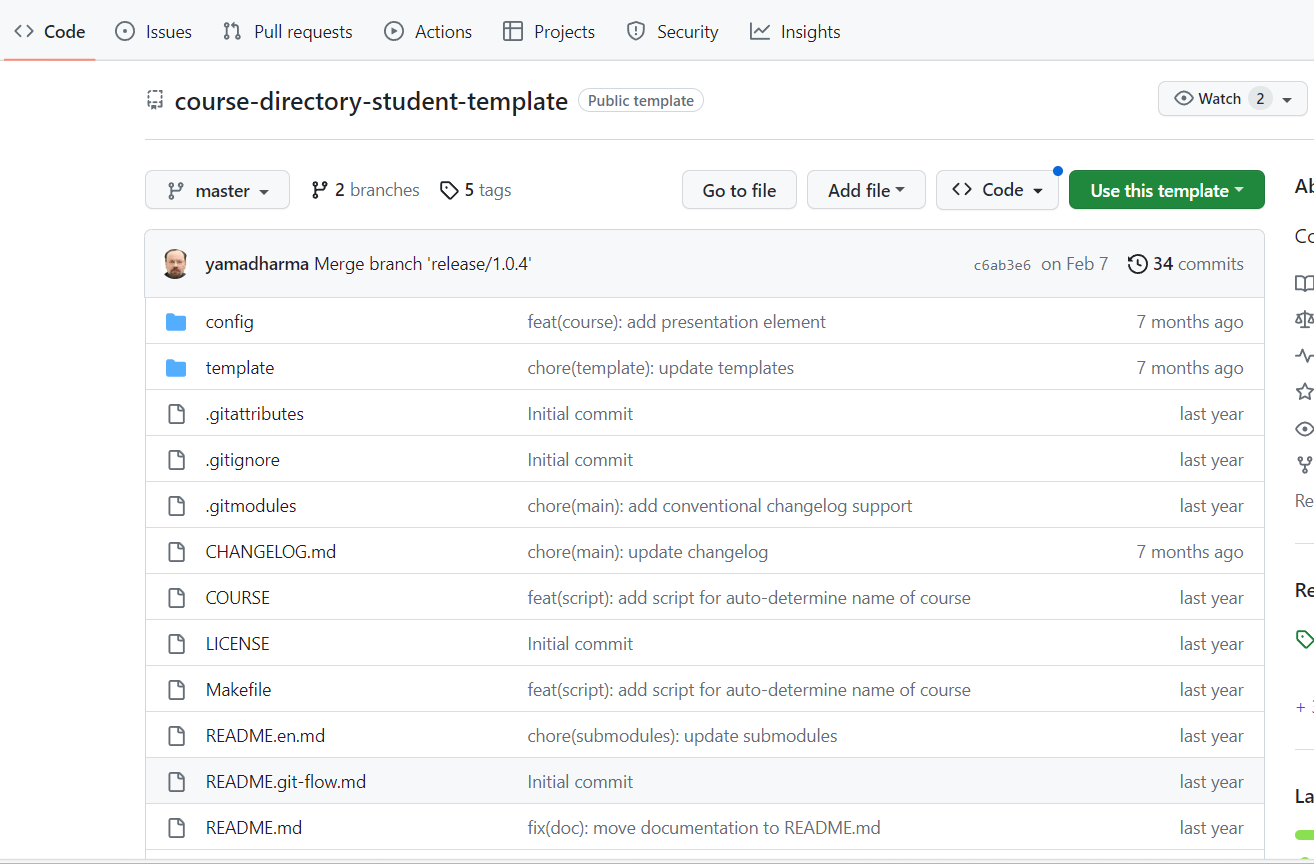
**6) Сознание рабочего пространства и репозитория курса на основе шаблона**

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.

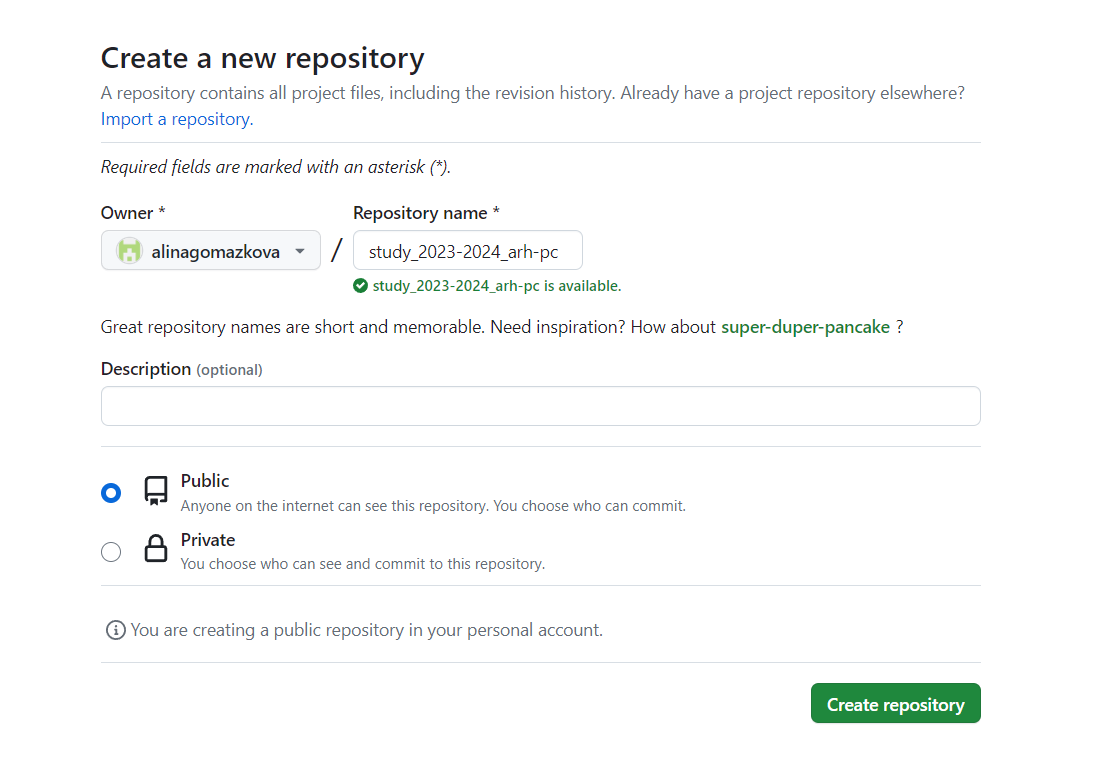


**7) Сознание репозитория курса на основе шаблона**

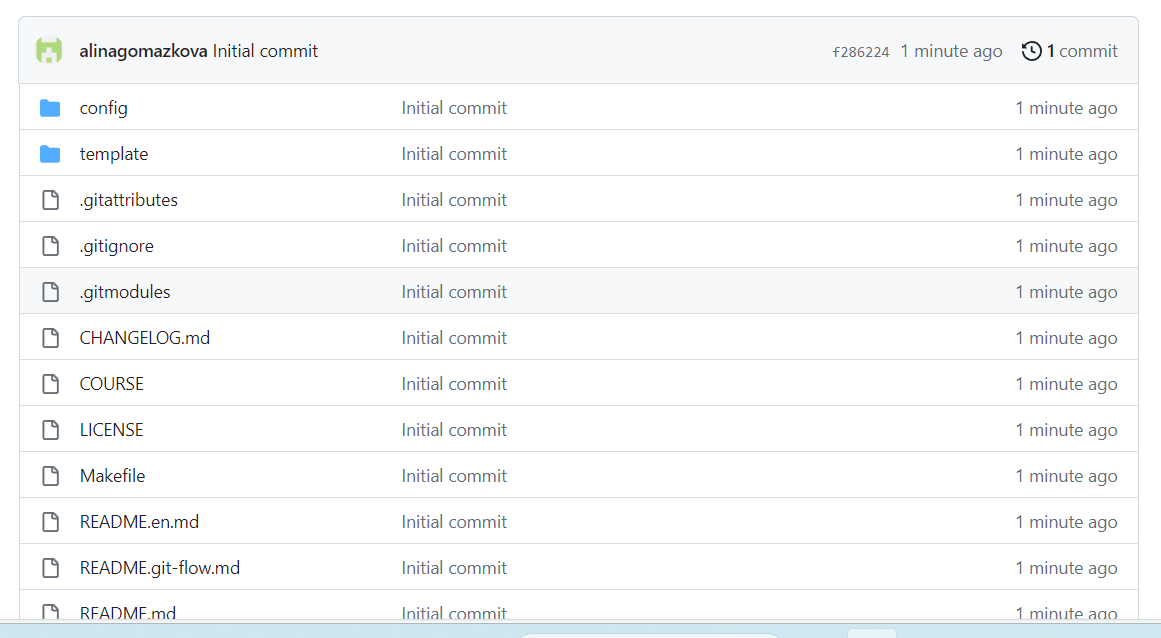
Репозиторий на основе шаблона можно создать через web-интерфейс github. Перейдите на станицу репозитория с шаблоном курса https://github.com/yamadharma/cour se-directory-student-template. Далее выберите Use this template.



В открывшемся окне задайте имя репозитория (Repository name) study\_2023-2024\_arh-pc и создайте репозиторий (кнопка Create repository from template).



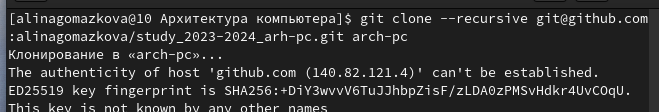
Репозиторий создан



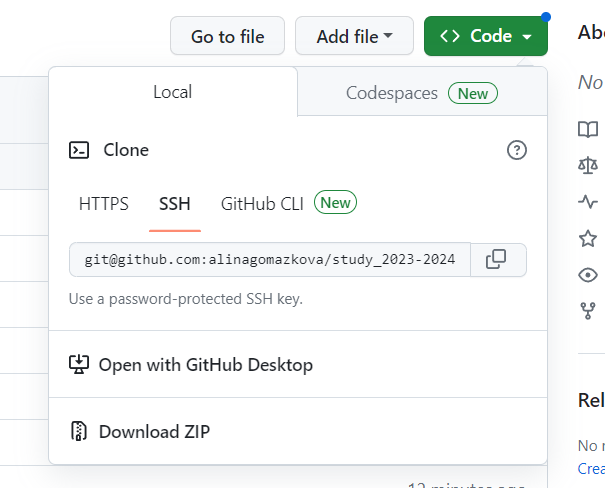
Через терминал перехожу в созданный каталог курса с помощью утилиты cd



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

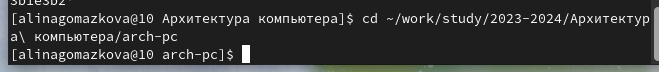


Копирую ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH».



**8) Настройка каталога курса**

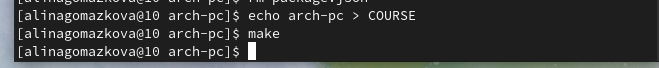
Перехожу в каталог arch-pc с помощью утилиты cd



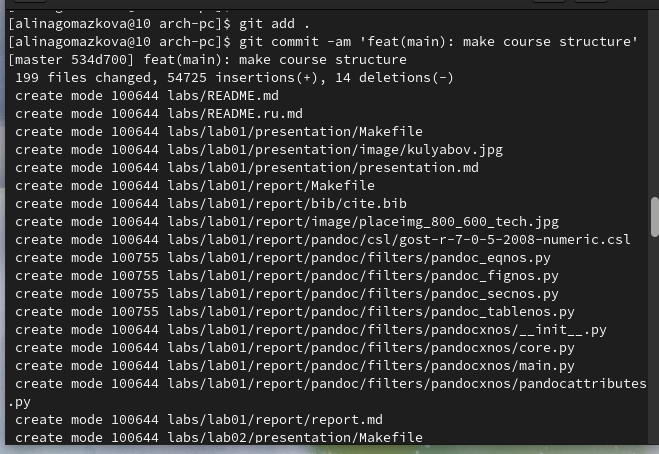
Удаляю лишние файлы с помощью утилиты rm



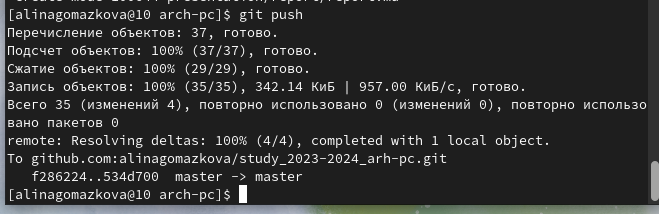
Создаю необходимые каталоги



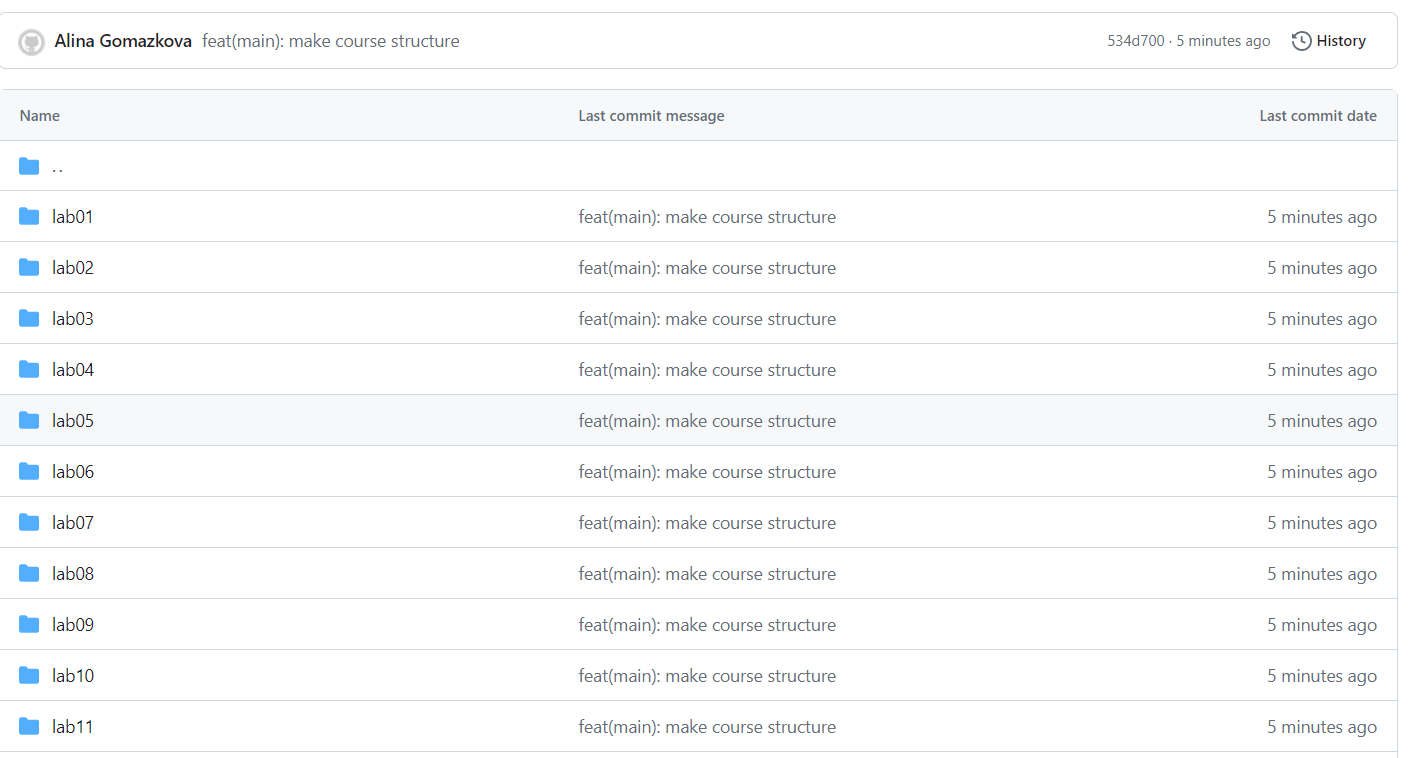
Отправляю созданные каталоги с локального репозитория на сервер: добавляю все созданные каталоги с помощью git add, комментирую и сохраняю изменения на сервере как добавление курса с помощью git commit



Отправляю все на сервер с помощью push



Проверяю правильность выполнения работы сначала на самом сайте GitHub



**9) Вывод**

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

**10) Список литературы**

1. [Архитектура ЭВМ (rudn.ru)](https://esystem.rudn.ru/pluginfile.php/2089082/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%962.%20%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F%20%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9%20Git.pdf)