**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**по лабораторной работе № 2**

*дисциплина: Архитектура компьютера*

Студентка: Зайцева У. В.

Группа: НКАбд-03-23

**МОСКВА**

2023 г.

**Содержание**

**1 Цель работы 3 2 Задание 4**

**3 Теоретическое введение 5**

**4 Выполнение лабораторной работы 6**

**5 Выводы 17**

**6 Список литературы**

**1 Цель работы**

Изучение идеологии и применения средств контроля версий. Приобрести практические навыки по работе с системой git.

1. **Задание**
2. **Настройка GitHub.**
3. **Базовая настройка Git.**
4. **Создание SSH-ключа.**
5. **Создание рабочего пространства и репозитория курса на основе шаблона.**
6. **Создание репозитория курса на основе шаблона.**
7. **Настройка каталога курса.**
8. **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 (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.

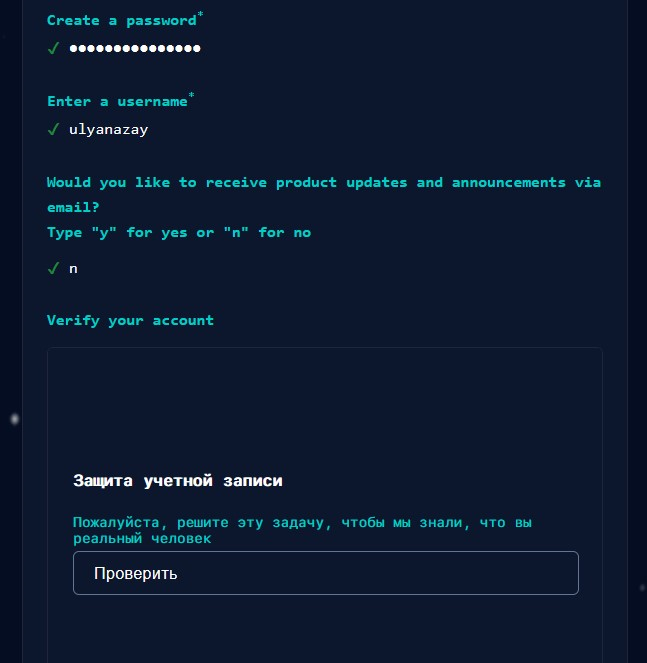
В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых —

Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

**4 Выполнение лабораторной работы**

**4.1. Настройка Github**

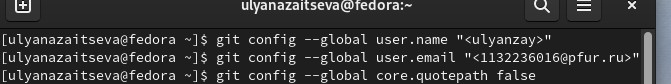
Создаю учётную записать на github и заполняю основные данные(рис. 1)



*Рис. 1. Заполнение данных учетной записи GitHub*

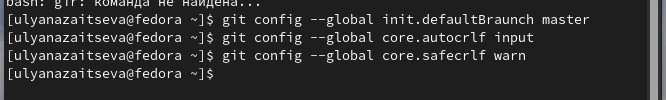
**4.2. Базовая настройка git**

Открываю терминал и ввожу команды, указывая своё имя и email. Настраиваю utf-8(рис. 1)



*Рис. 2. Предварительная конфигурация git и настройка кодировки*

Задаю имя начальной ветки(master), использую параметры autocrlf и safecrlf(рис. 3)

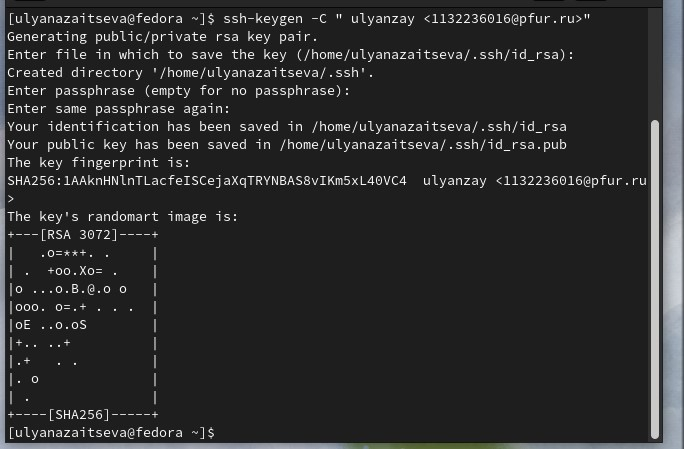


*Рис. 3. Создание имени для начальной ветки и использование параметров*

**4.3. Создание SSH-ключа**

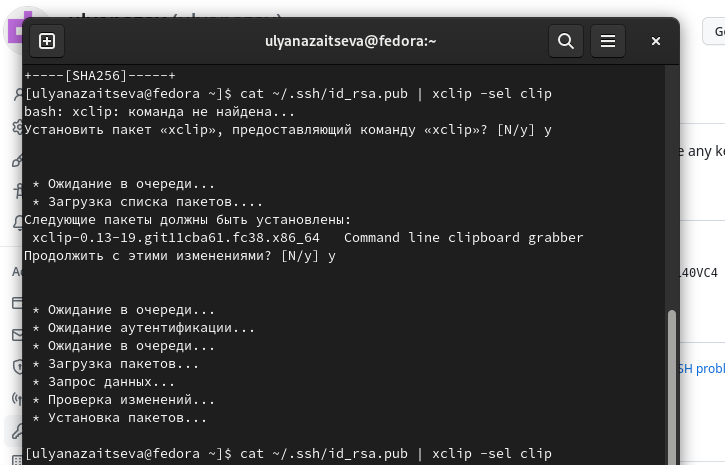
Для последующей идентификации пользователя на сервере репозиториев необходимо

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)

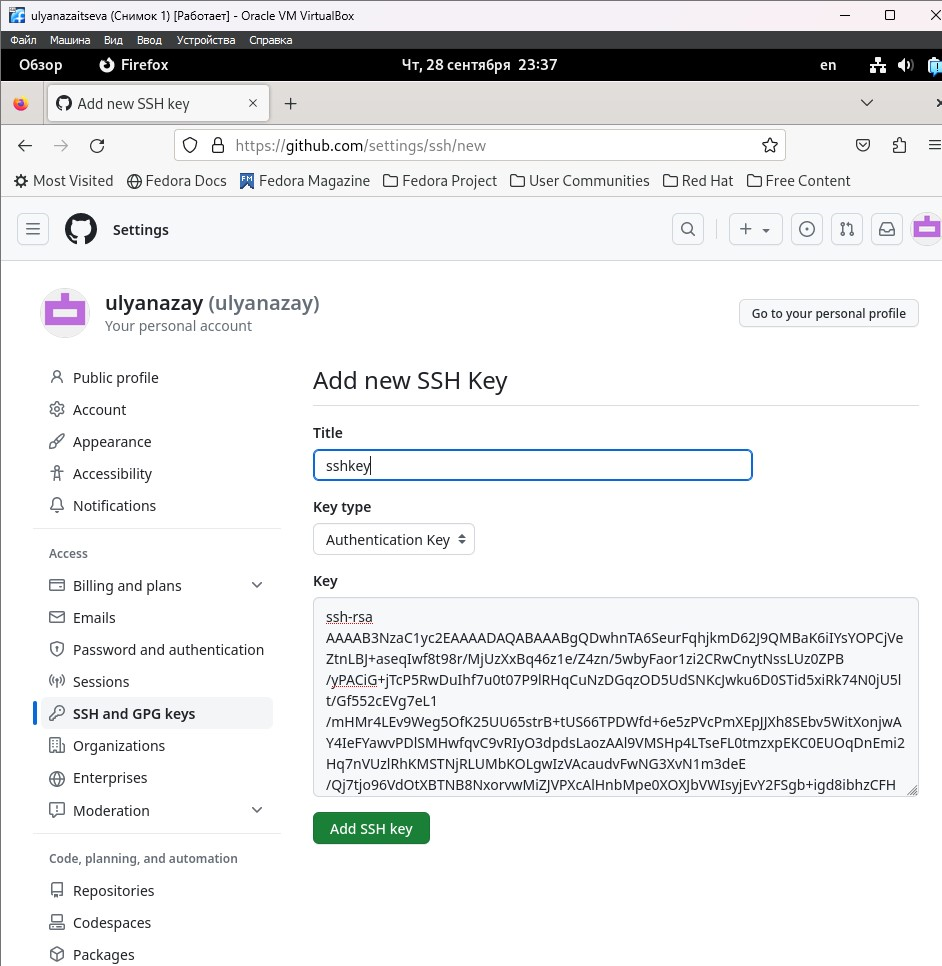


*Рис. 4. Генерация SSH-ключа*

Xclip – утилита, которая позволяет скопировать любой текст через терминал. Устанавливаю необходимый пакет для данной утилиты уже при попытке её использовать. После установки заново пишу команду и копирую содержимое файла(сгенерированный ssh-ключ)(рис. 5)



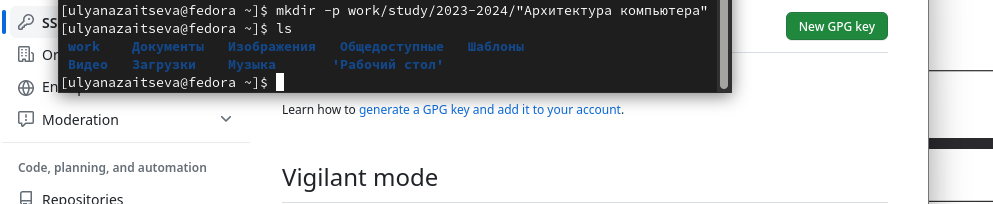
*Рис. 5. Установка утилиты xclip и копирование содержимого файла*

Вставляю скопированный ключ в поле Key, задаю имя ключу в поле Title(рис. 6)

*Рис. 6. Добавление ключа*

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

Возвращаюсь в терминал и создаю рабочее пространство “Архитектура компьютера” c помощью команды mkdir. С помощью команды ls проверяю выполнение команды(рис. 7)



*Рис. 7. Создание рабочего пространства*

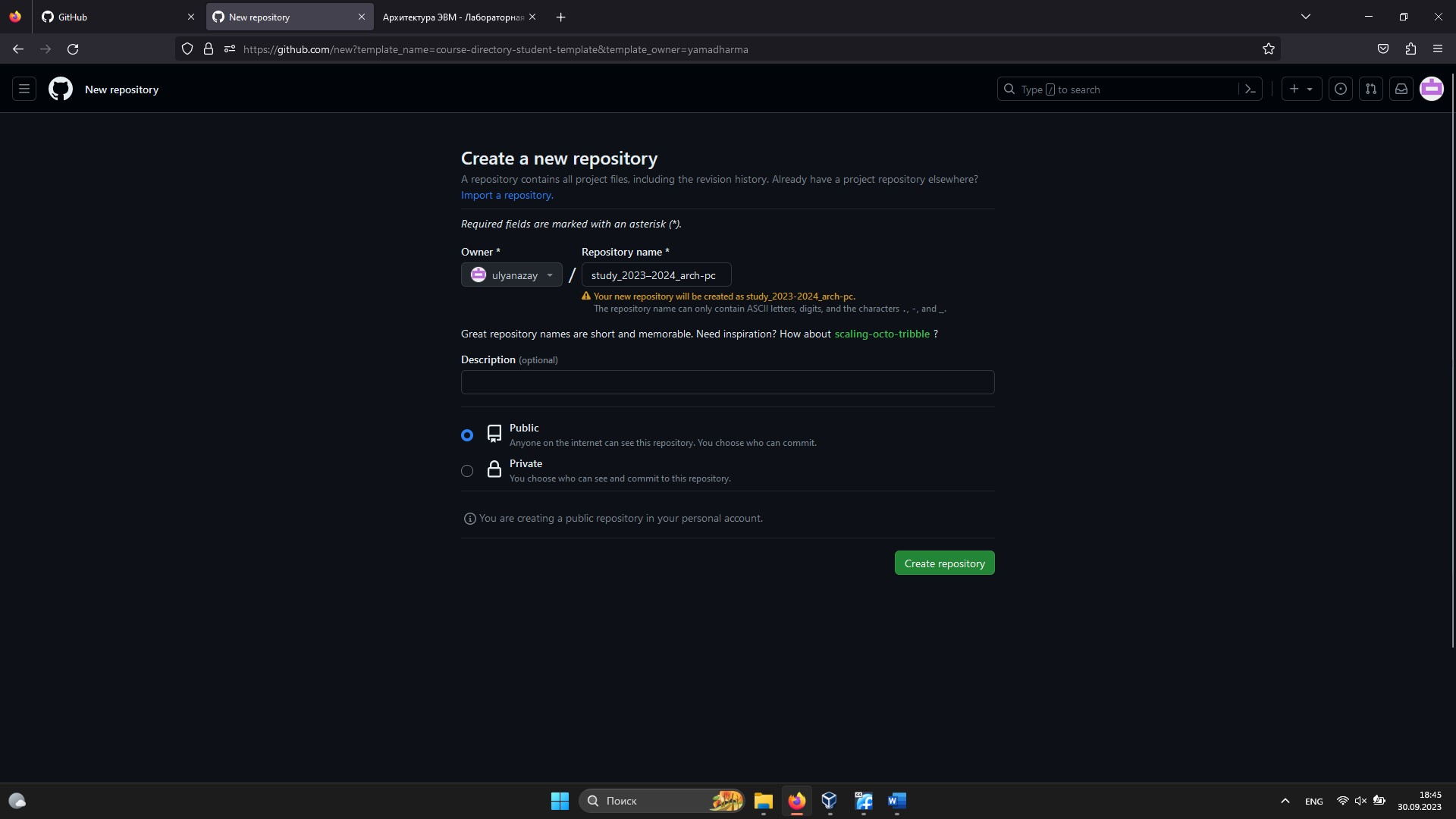
**4.5. Создание репозитория курса на основе шаблона**

В браузере перехожу на страницу репозитория с шаблоном курса https://github.com/yamadharma/course-directory-student-template. Выбираю «Use this template», чтобы использовать шаблон для своего репозитория(рис. 8)



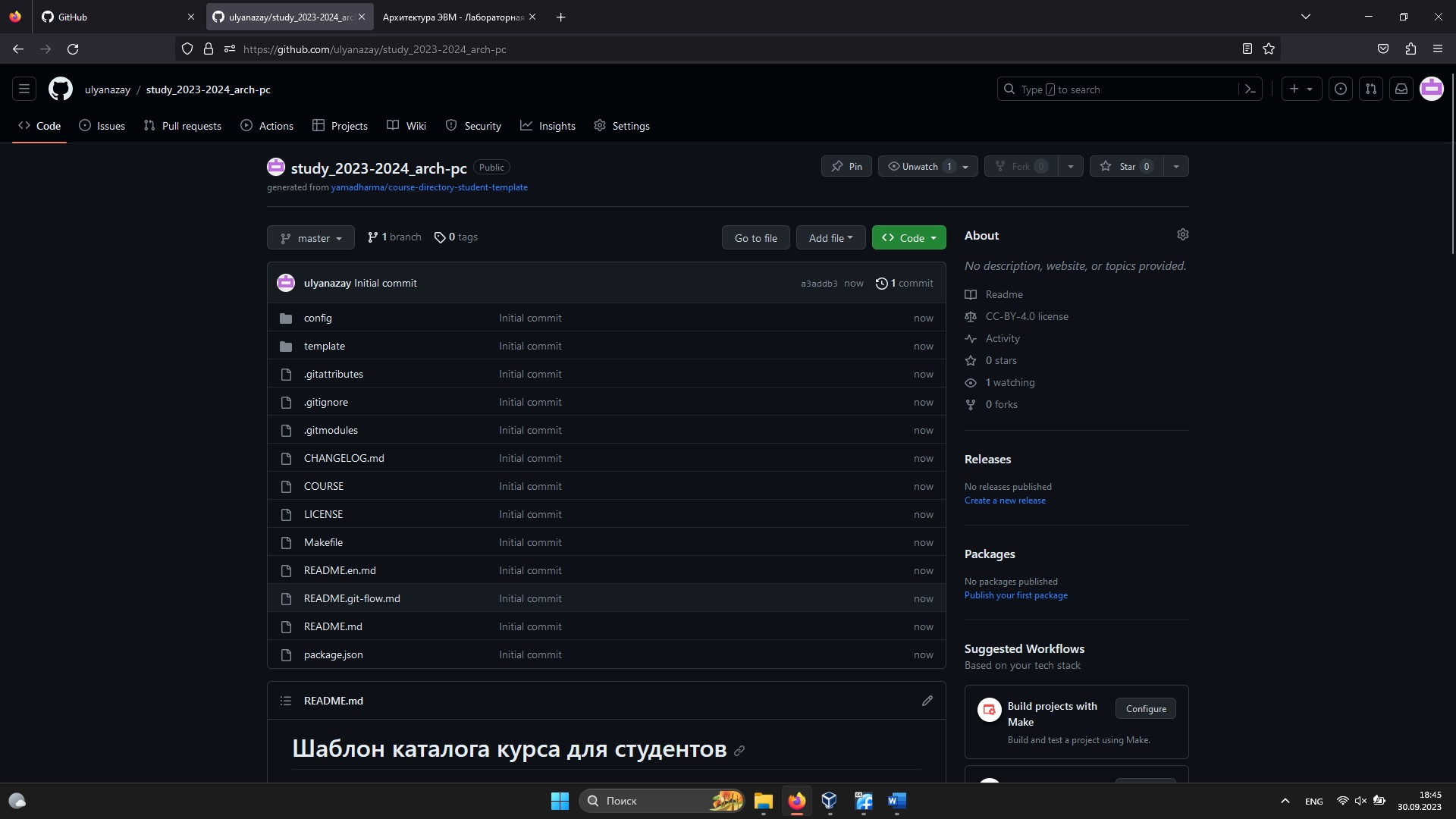
*Рис. 8. Страница шаблона для репозитория*

Задаю имя репозитория study\_2023–2024\_arсh-pc и создаю репозиторий (кнопка Create repository)(рис. 9)



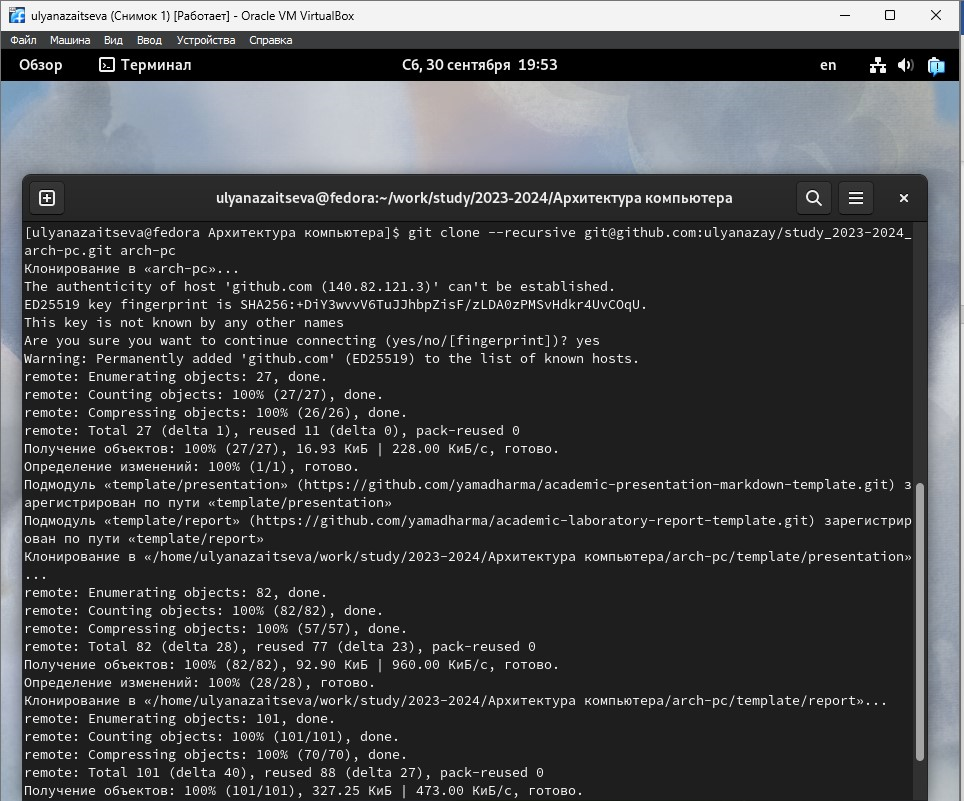
*Рис. 9. Окно создания репозитория*

Репозиторий создан!(рис. 10)



*Рис. 10. Созданный репозиторий*

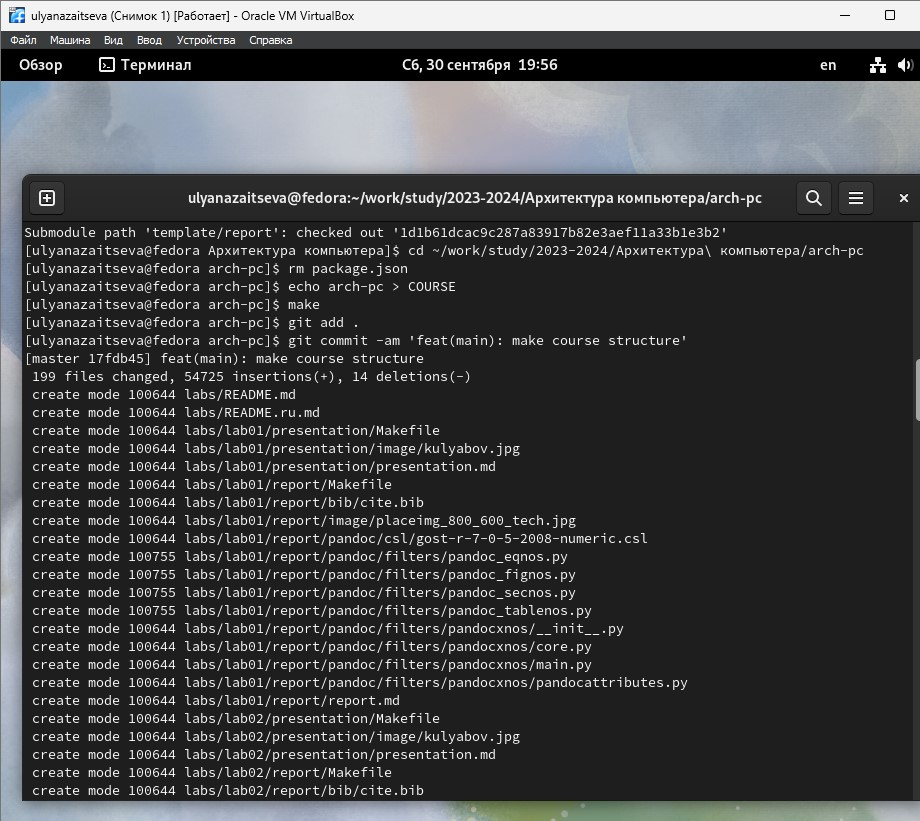
Открываю терминал и перехожу в каталог курса: cd ~/work/study/2023-2024/"Архитектура компьютера"(плохо обрезала скрин и данная команда в него не попала, но по полю перед знаком $ как раз указан каталог в котором я нахожусь, а именно каталог “Архитектура компьютера”, в который я и перешла до клонирования прописав данную команду!). Клонирую созданный репозиторий.(рис. 11)



*Рис. 11. Клонирование репозитория*

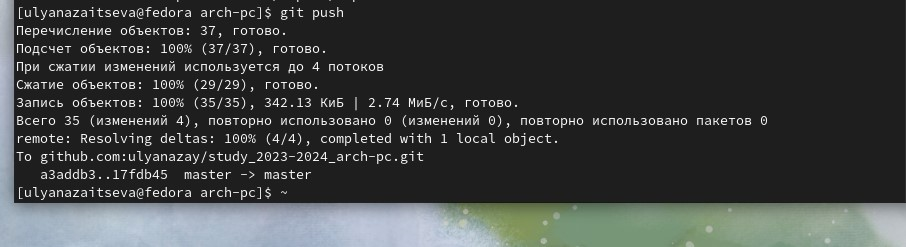
**4.6. Настройка каталога курса**

Перехожу в каталог курса с помощью команды cd, удаляю лишние файлы с помощью команды rm, создаю необходимые каталоги, добавляю их и сохраняю изменения на сервере(рис. 12)



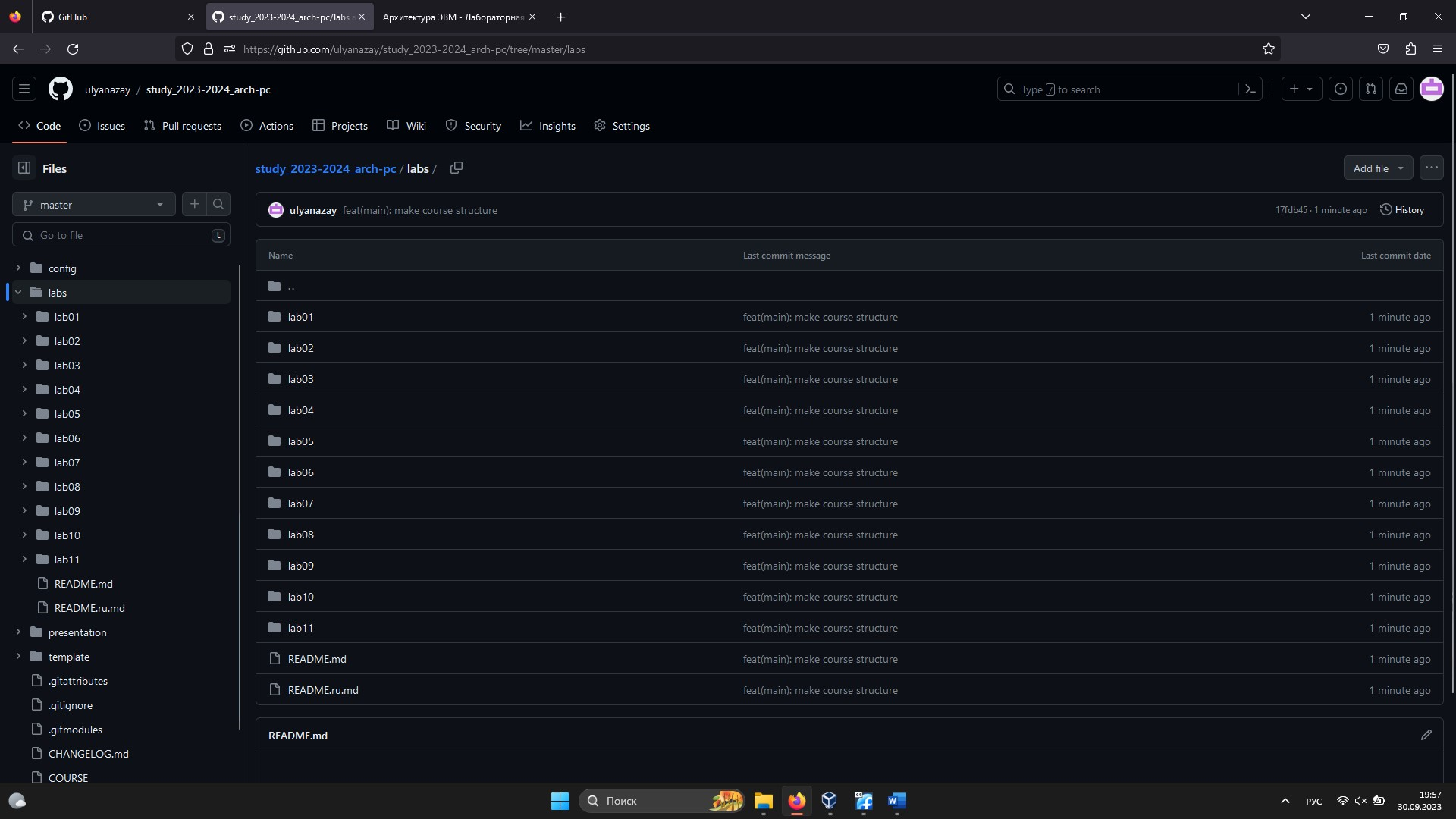
*Рис. 12. Перемещение между директориями, удаление лишних файлов, создание каталогов, сохранение изменений*

Отправляю созданные каталоги на сервер(рис. 13)



*Рис. 13. Выгрузка изменений на сервер*

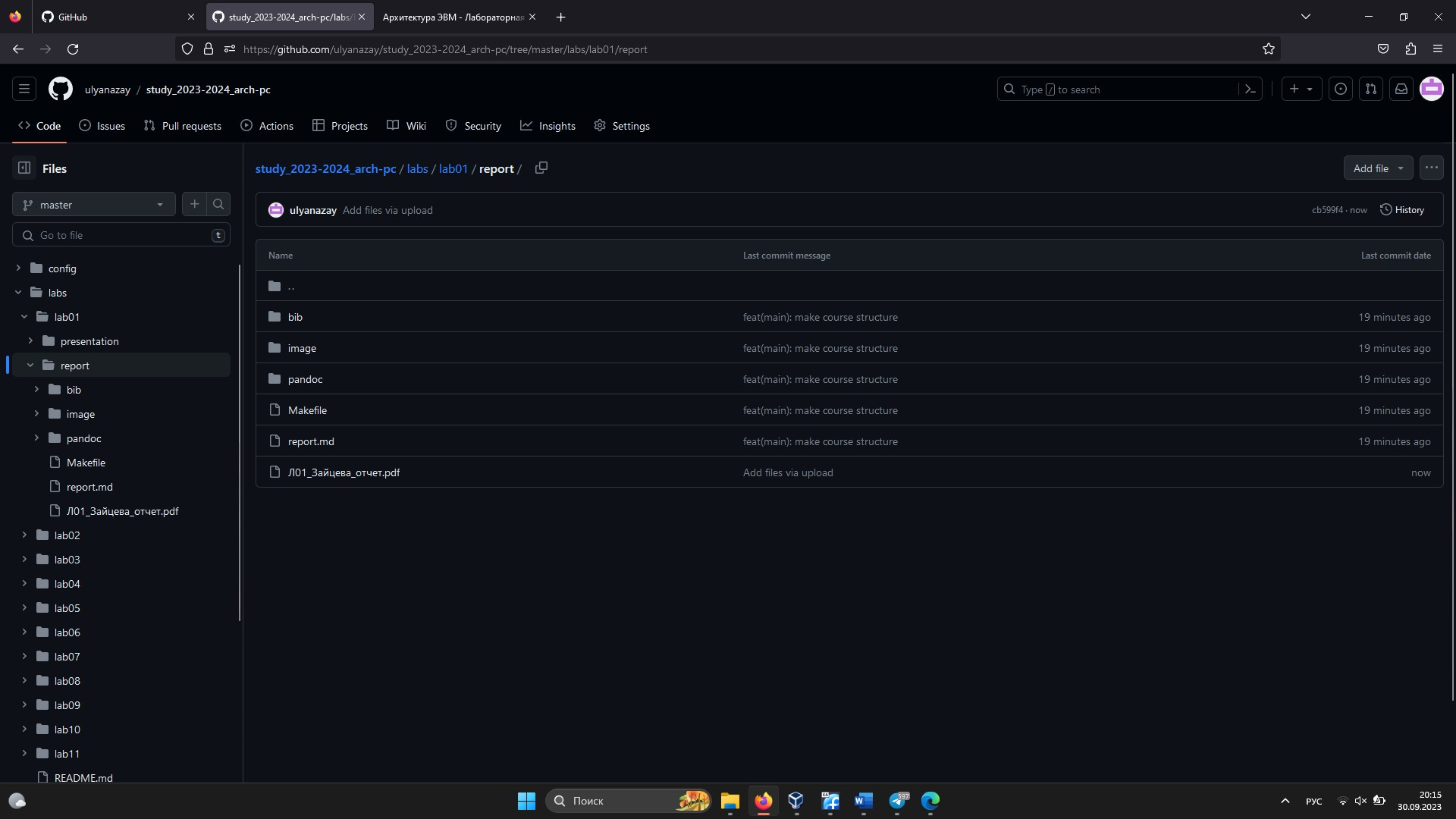
Проверяю правильность выполнения работы на сайте GitHub (рис. 14).



*Рис. 14. Страница репозитория*

**4.7. Выполнение заданий для самостоятельной работы**

Загружаю отчёт по первой лабораторной работе в нужный каталог, а именно lab01/report, используя сайт(рис. 15).



*Рис. 15. Каталог lab01/report*

Когда завершу данный отчёт, загружу его в 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(esystem.rudn.ru)