**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**по лабораторной работе № 2**

*дисциплина: Архитектура компьютера*

Студент: Шабакова Карина Баировна

с\б:1132242469

Группа:НКАбд-05-24

**МОСКВА**

1. г.

**Содержание**

**1 Цель работы**

**2 Задание**

**3 Теоретические введение**

**4 Выполнение лабораторной работы**

4.1 Настройка GitHub

4.2 Базовая настройка Git

4.3 Создание SSH-ключа

4.4 Создание рабочего пространства и репозитория курса на

основе шаблона

4.5 Создание репозитория курса на основе шаблона

4.6 Настройка каталога курса

4.7 Выполнение заданий для самостоятельной работы

**5 Выводы**

**6 Список литературы**

**1 Цель работы**

Целью работы является изучить идеологию и применение средств контроля версий. 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 (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 учетную запись на сайте GitHub и заполнила основные данные.

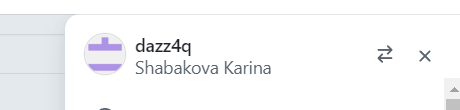


рис 1 Аккаунт GitHub

**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. 2: Предварительная конфигурация git

Настраиваю utf-8 в выводе сообщений git для корректного отображения символов.



Рис. 3: Настройка кодировки

Задаю имя «master» для начальной ветки



Рис. 4: 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. CR и LF – это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах.



Рис. 5: Параметр autocrlf

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.



Рис. 6: 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](mailto:work@email)”, указывая имя владельца и электронную почту владельца . Ключ автоматически сохранится в каталоге ~/.ssh/.

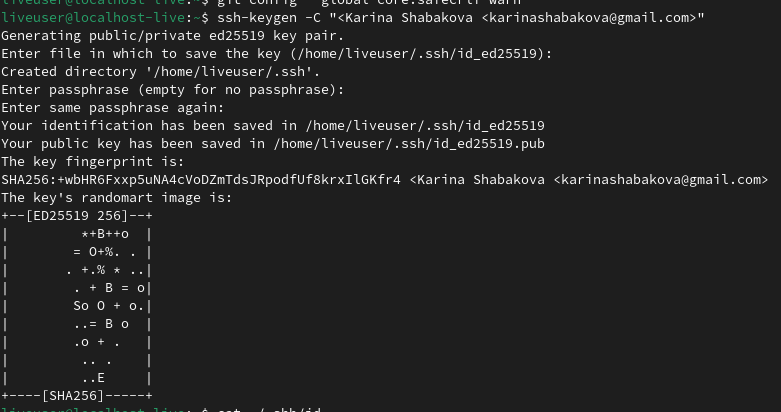


Рис. 7: Генерация SSH-ключа

Через комнаду cat выводит ключ, копирую



Рис. 8: Копирование содержимого файла

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

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

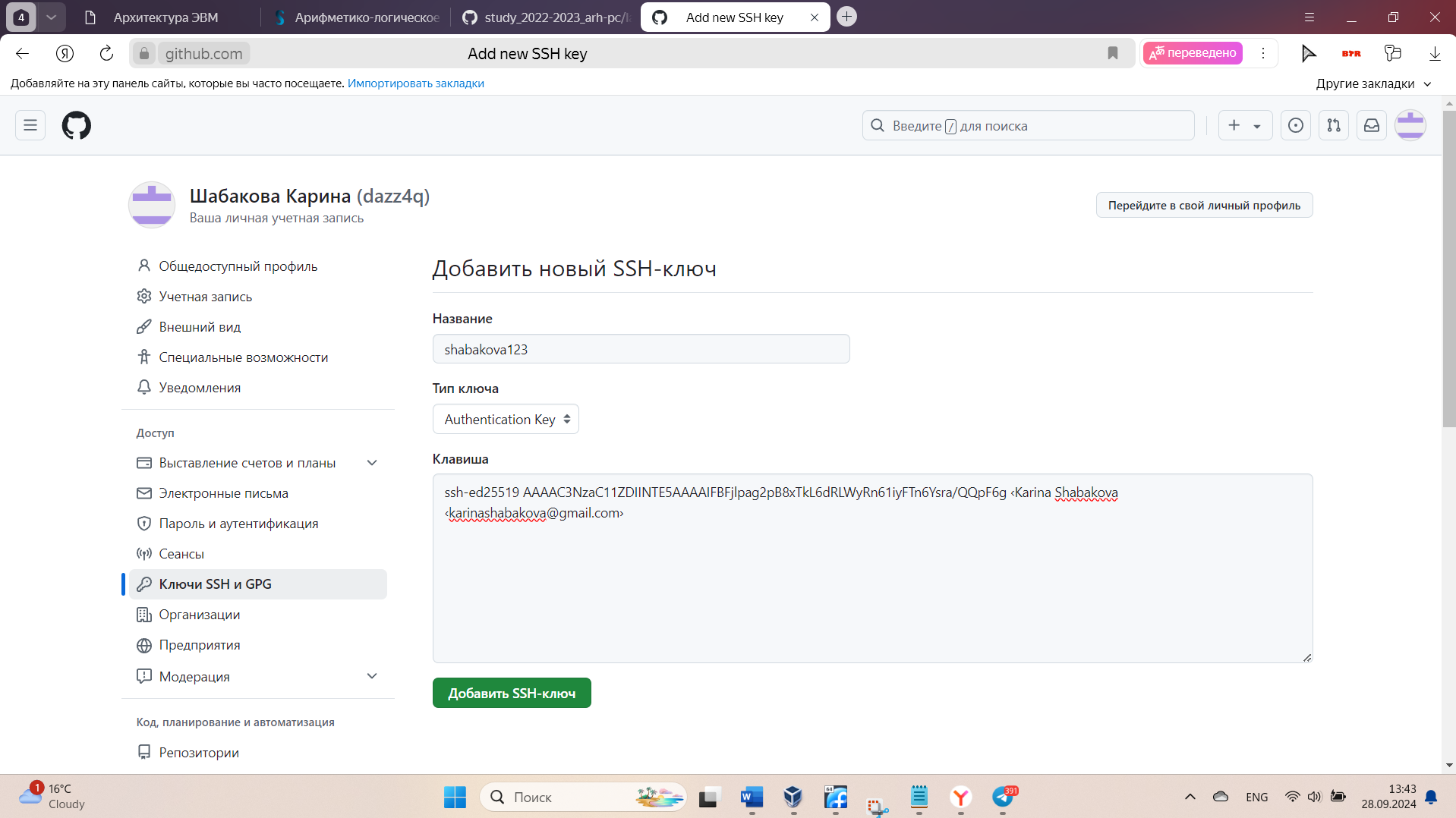


Рис. 9: Добавление ключа

**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/2022-2023/“Архитектура компьютера” рекурсивно. Далее проверяю с помощью ls, действительно ли были созданы необходимые мне каталоги

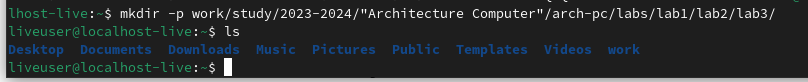


Рис. 10: Создание рабочего пространства

**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

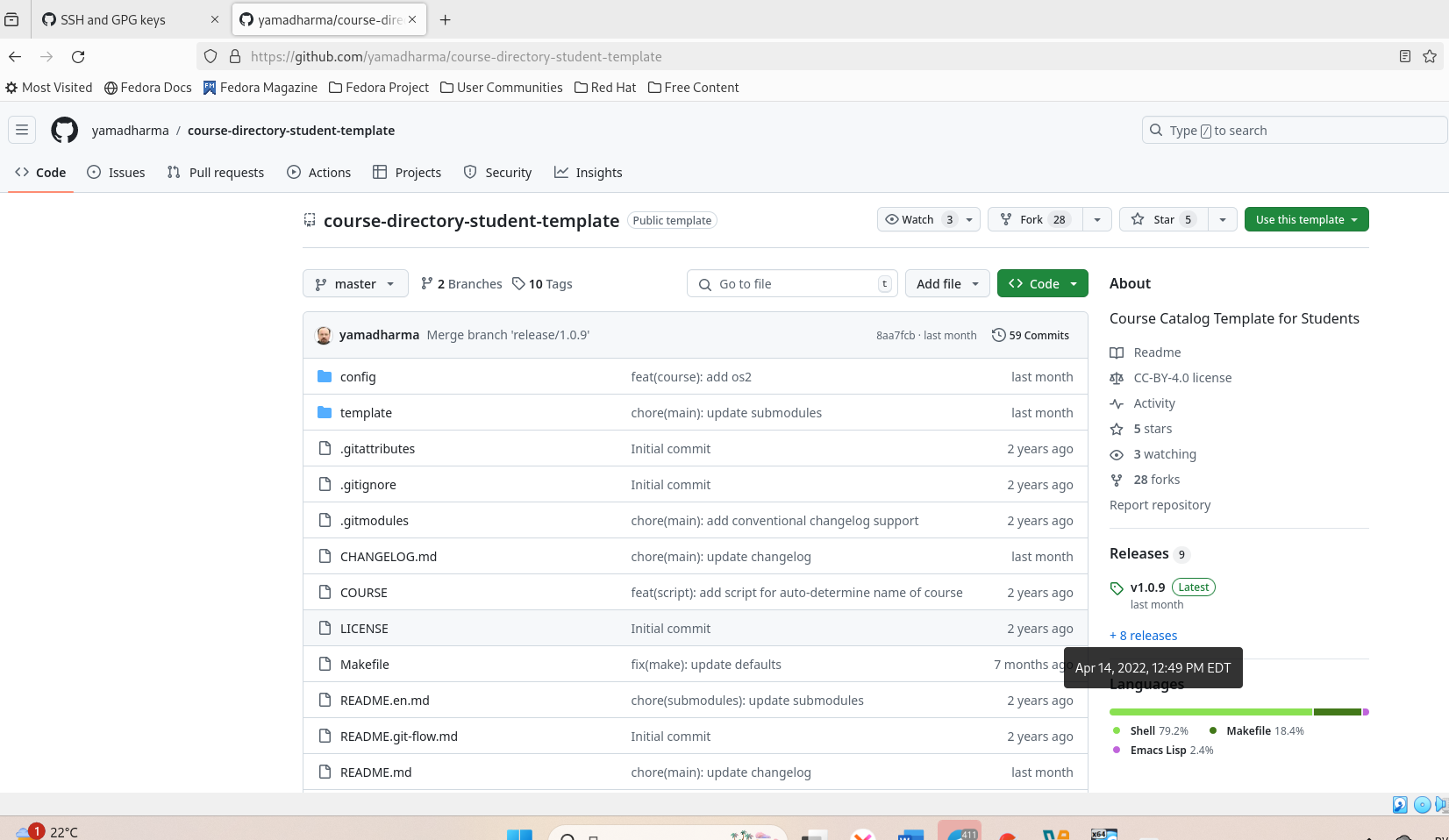
****

Рис. 11: Страница шаблона для репозитория

В открывшемся окне задаю имя репозитория (Repository name): study\_2022–2023\_arh-pc и создаю репозиторий, нажимаю на кнопку «Create repository from template»

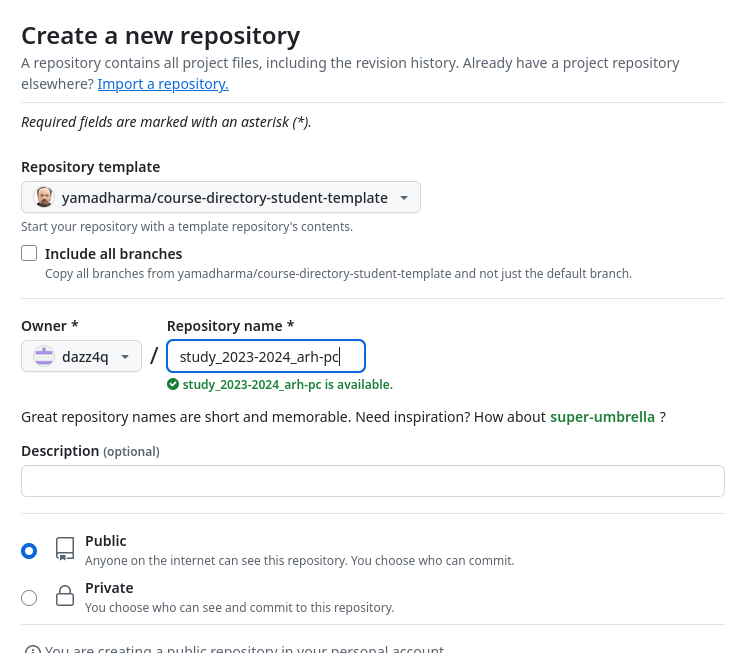
****

Рис. 12: Окно создания репозитория

Репозиторий создан

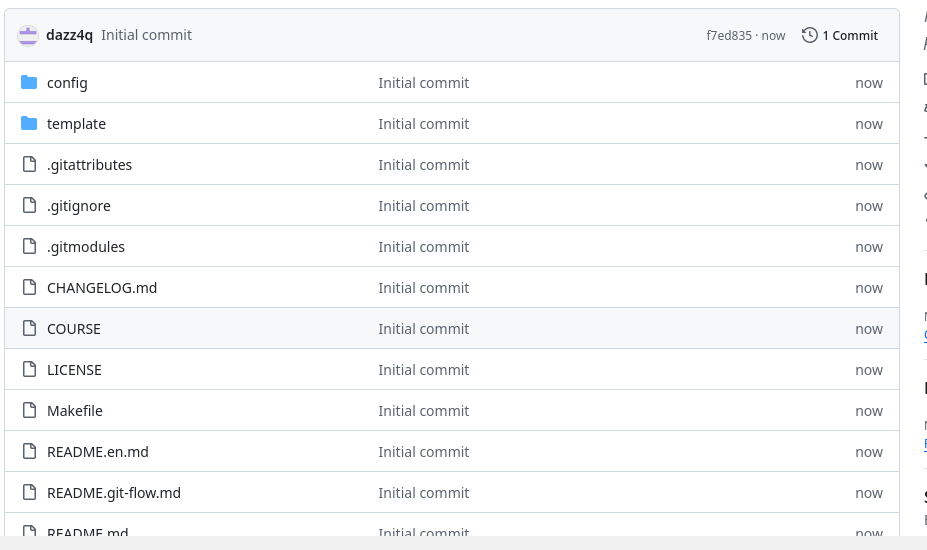
****

Рис. 13: Созданный репозиторий

Через терминал перехожу в созданный каталог курса с помощью утилиты cd

****

Рис. 14: Перемещение между директориями

Клонирую созданный репозиторий с помощью команды git clone –recursive git@github.com:/study\_2022–2023\_arh-pc.git arch-pc

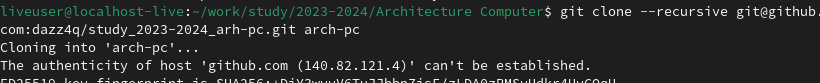
****

Рис. 15: Клонирование репозитория

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

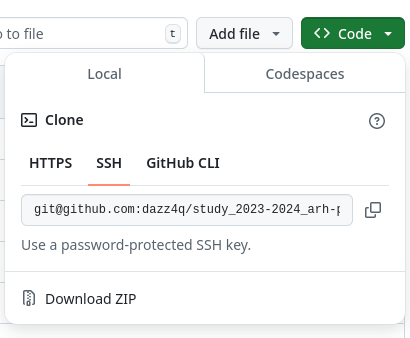
****

Рис. 16: Окно с ссылкой для копирования репозитория

**4.6. Настройка каталога курса**

Перехожу в каталог arch-pc с помощью утилиты cd

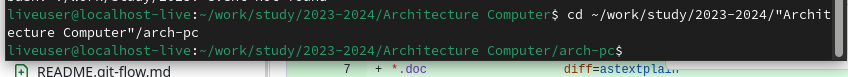
****

Рис. 17: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты rm

****

Рис. 18: Удаление файлов

Устанавила команду make

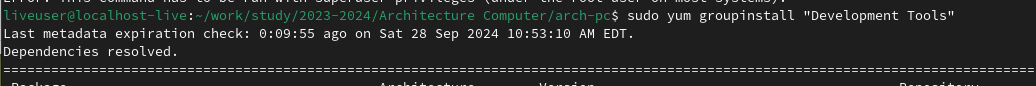
****

Рис. 19: установка make

Создаю необходимые каталоги, проверила командой ls созданные каталоги

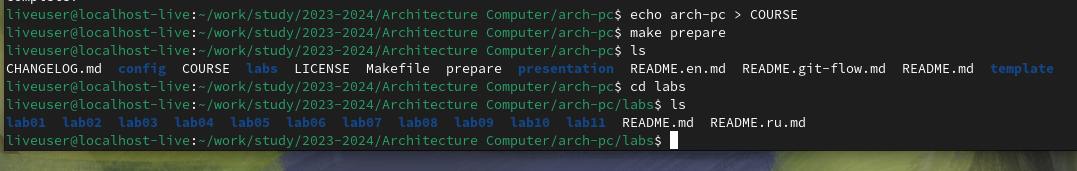
****

Рис. 20: 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

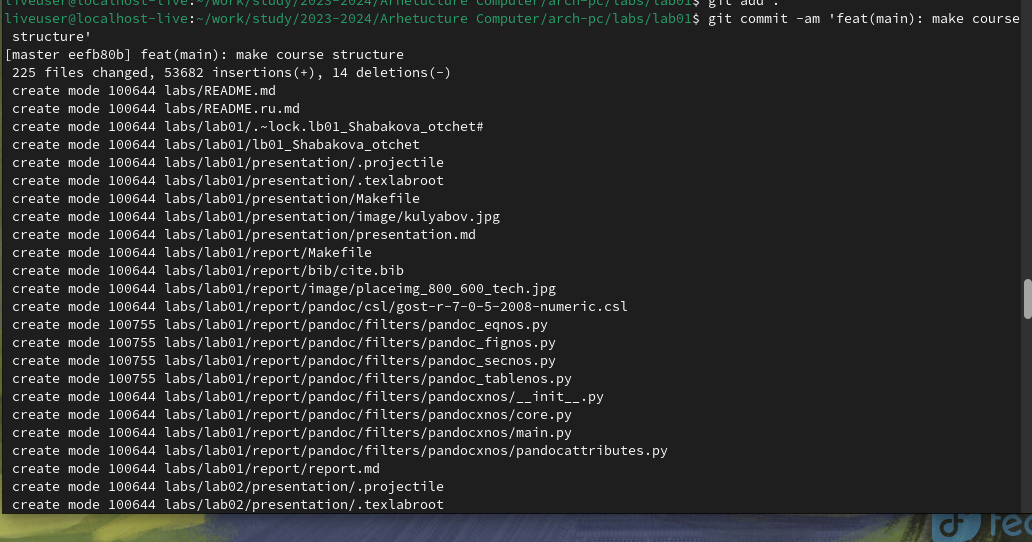
****

Рис. 21: Добавление и сохранение изменений на сервере

Отправляю все на сервер с помощью push

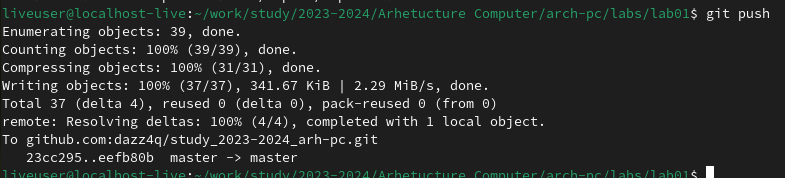
****

Рис. 22: Выгрузка изменений на сервер

Проверила правильность создания иерархии рабочего пространства в локальном репозитории на странице github

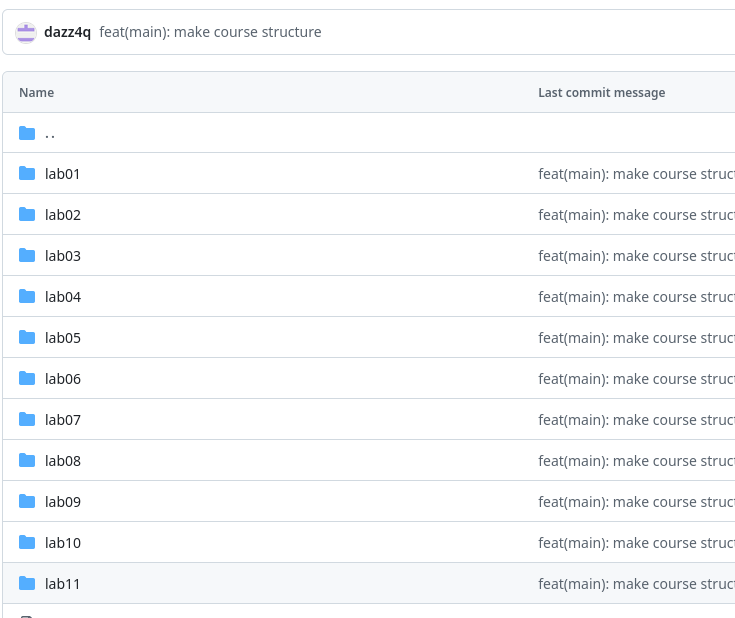


Рис 23: выполнение лабораторной

## 4.7 Выполнение заданий для самостоятельной работы

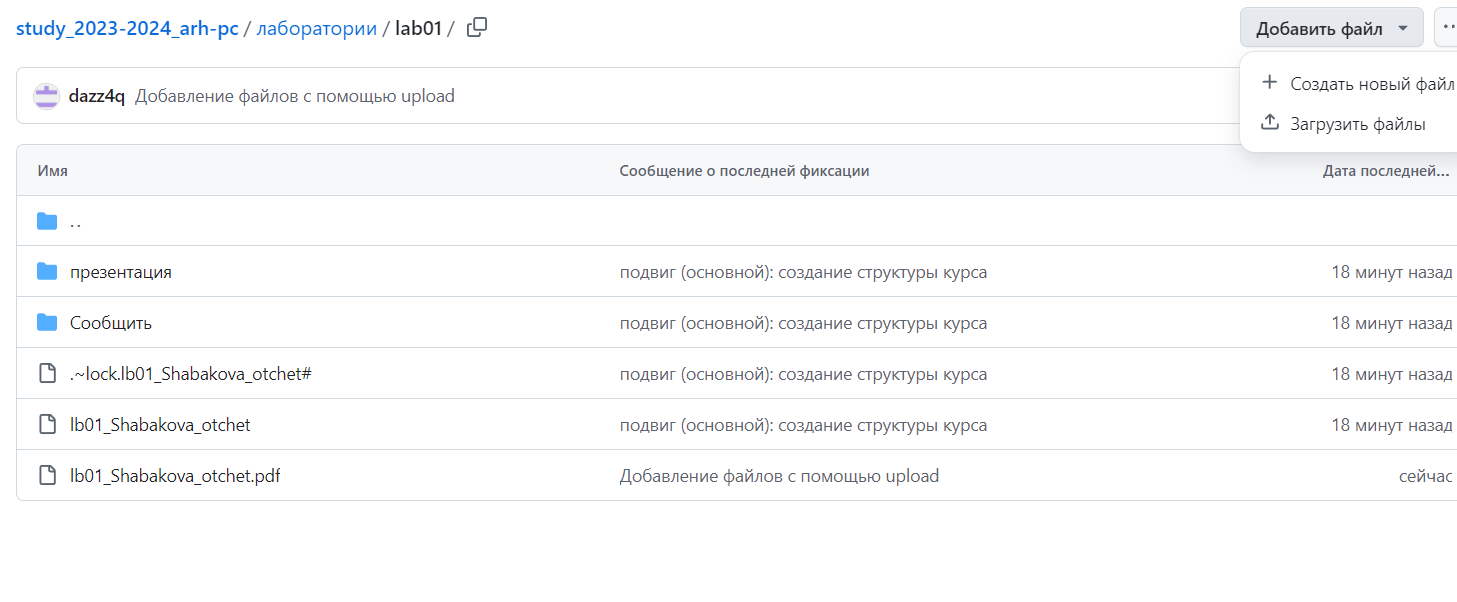
Создаю отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs>lab02>report).

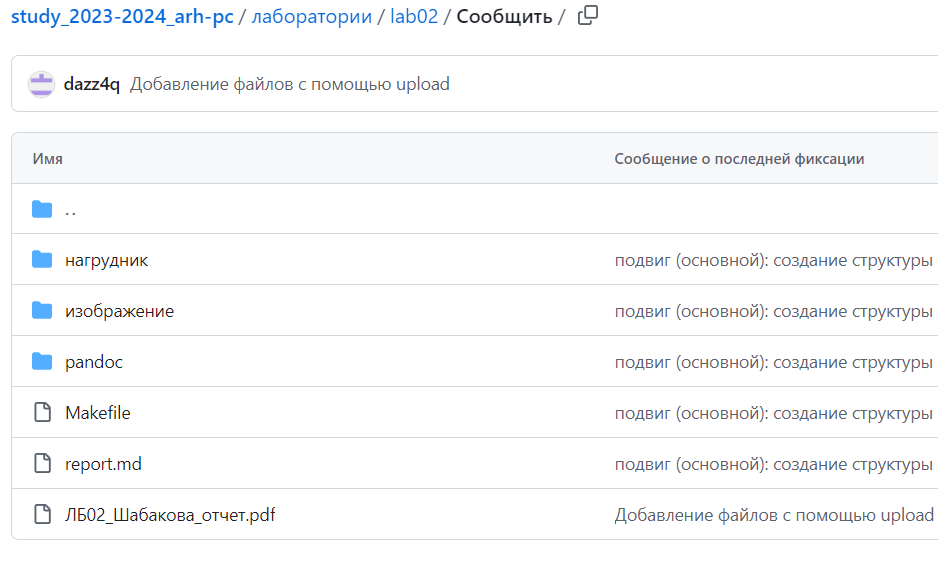




Рис 24: создание файлов

Загрузила на github файлы лабораторных работ

****

****

*Рис 25: загрузка файлов*

# **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. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.

2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.

3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander. org/.

4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.

5. Newham C. Learning the bash Shell: Unix Shell Programming. — O’Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.

6. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.

7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.

8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.

9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.

10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.

11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.

12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.

13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.

14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: <http://www.stolyarov.info/books/asm_unix>.

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).

16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).