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:

Жарикова Таисия Александровна

Группа:

НКАбд-05-24

МОСКВА

2024 г.

Содержание

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 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

Аккаунт GitHub у меня уже имеется. Сменила привязанную почту на корпоративную.

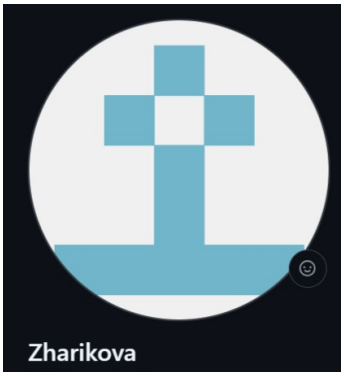


Рис. 4.1: Аккаунт GitHub

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).

```
liveuser@localhost-live:~$ git config --global user.name "<Taisia Zharikova>"
liveuser@localhost-live:~$ git config --global user.email "<1132247522@pfur.ru>"
```

Рис. 4.3: Предварительная конфигурация git

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

Рис. 4.4: Настройка кодировки

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

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.

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.

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/.

```
taizha@192:~$ ssh-keygen -C "Taisia Zharikova <1132247522@pfur.ru>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/taizha/.ssh/id_ed25519):
Created directory '/home/taizha/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/taizha/.ssh/id_ed25519
Your public key has been saved in /home/taizha/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:LNeW2xhQK2o+ghrnkCDpmcKehlOLCUXXu8enGiKNUuQ Taisia Zharikova <1132247522@
pfur.ru>
The key's randomart image is:
+--[ED25519 256]--+
|      .    .     |
|  . . .  . .     |
| ...   .o .      |
| +.   .o + .     |
|+.E   +oS =      |
|*o++ o.oo..=     |
|OO=.+ +. oo .    |
|BOo. o o.        |
|o+.   ..         |
+----[SHA256]-----+
```

Рис. 4.8: Генерация SSH-ключа

Xclip – утилита, позволяющая скопировать любой текст через терминал. Сначала её надо установить. Устанавливаю xclip с помощью команды dnf install, введя в начале команды sudo (рис. 4.9).

```
liveuser@localhost-live:~$ sudo dnf install xclip
Fedora 40 - x86_64                              2.7 MB/s |  20 MB     00:07
Fedora 40 openh264 (From Cisco) - x86_64        2.1 kB/s | 1.4 kB     00:00
Fedora 40 - x86_64 - Updates                    3.7 MB/s |  11 MB     00:02
Dependencies resolved.
================================================================================
 Package        Architecture    Version                     Repository       Size
================================================================================
Installing:
 xclip          x86_64          0.13-21.git11cba61.fc40     fedora           37 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 37 k
Installed size: 62 k
Is this ok [y/N]: y
Downloading Packages:
xclip-0.13-21.git11cba61.fc40.x86_64.rpm        414 kB/s |  37 kB     00:00
--------------------------------------------------------------------------------
Total                                           126 kB/s |  37 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : xclip-0.13-21.git11cba61.fc40.x86_64                   1/1
  Running scriptlet: xclip-0.13-21.git11cba61.fc40.x86_64                   1/1

Installed:
  xclip-0.13-21.git11cba61.fc40.x86_64

Complete!
```

Рис. 4.9: Установка утилиты xclip

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

```
liveuser@localhost-live:~$ cat ~/.ssh/id_ed25519.pub | xclip -sel clip
```

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).

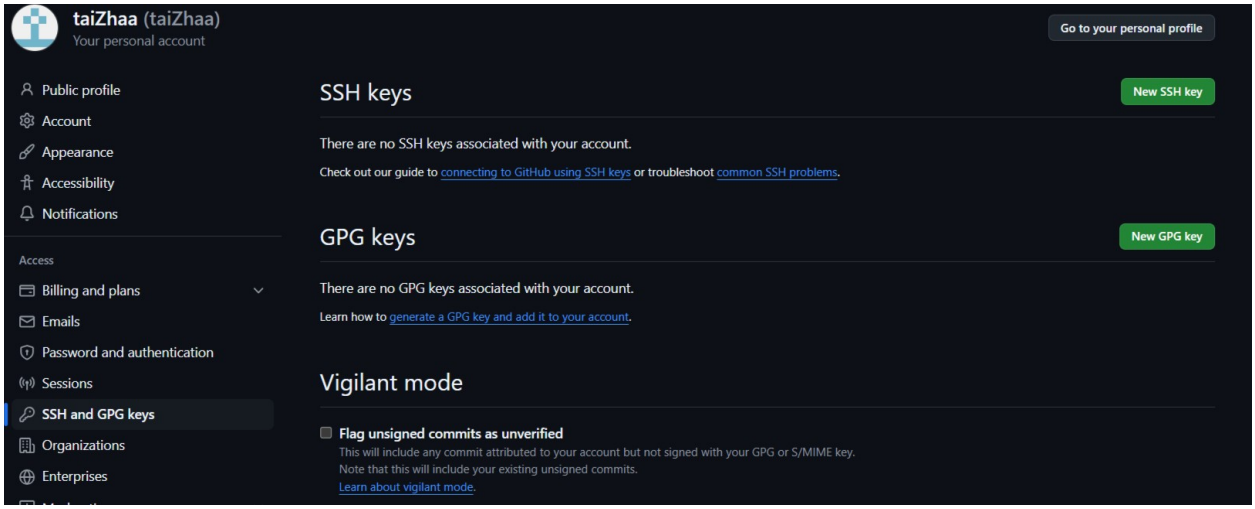


Рис. 4.11: Окно SSH and GPG keys

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

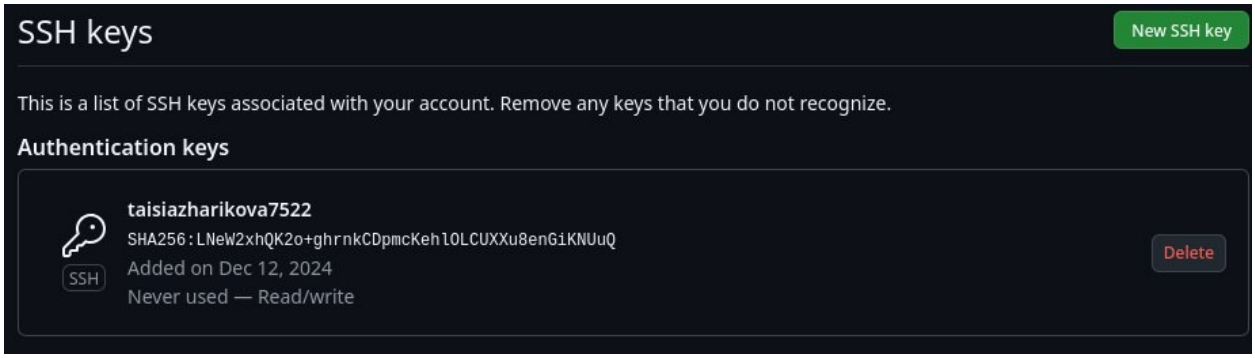


Рис. 4.12: Добавление ключа

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/2024-2025/“Арх пк” рекурсивно. Далее проверяю с помощью ls, действительно ли были созданы необходимые мне каталоги (рис. 4.13).

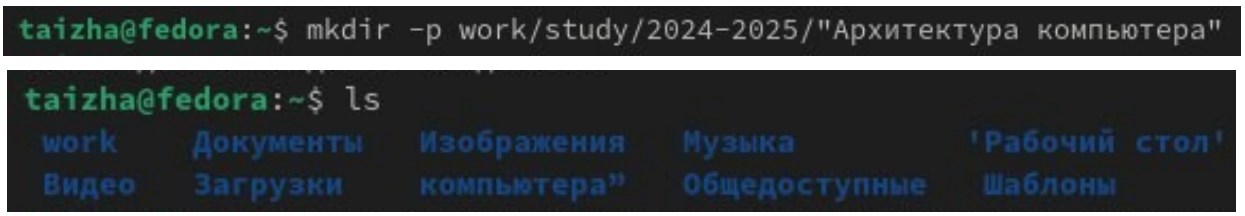


Рис. 4.13: Создание рабочего пространства

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).

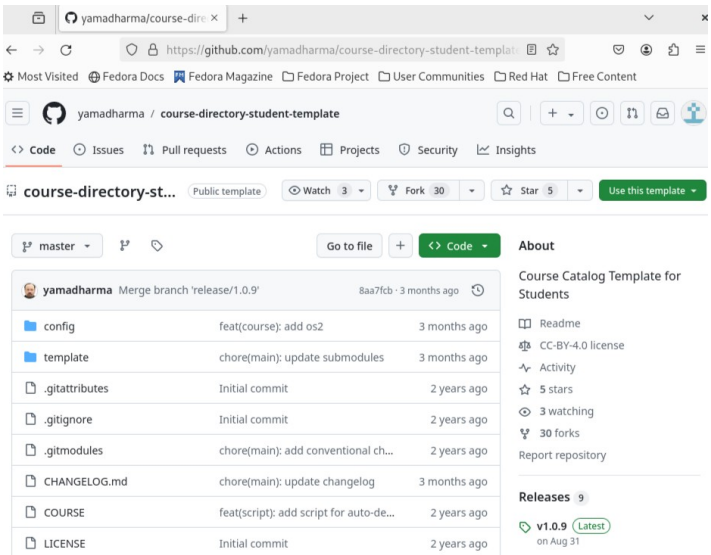


Рис. 4.14: Страница шаблона для репозитория

В открывшемся окне задаю имя репозитория (Repository name): study_2024–2025_arhpc и создаю репозиторий, нажимаю на кнопку «Create repository from template» (рис. 4.15).

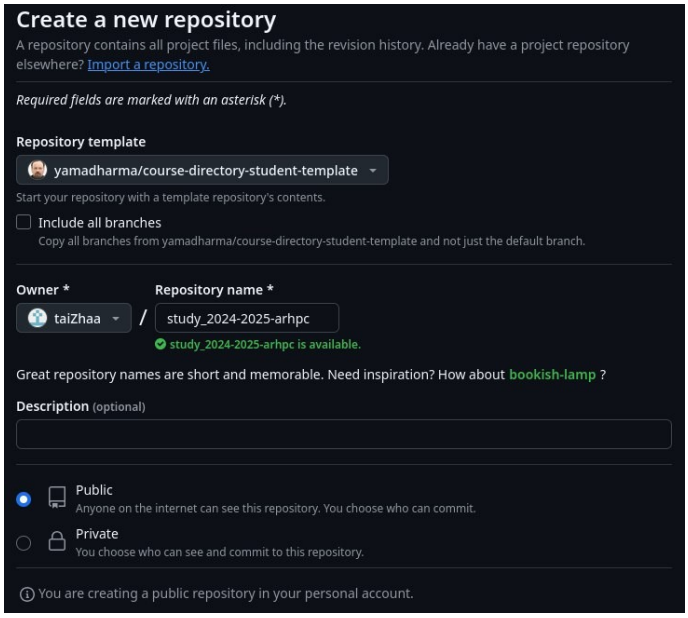


Рис. 4.15: Окно создания репозитория Репозиторий создан (рис. 4.16).

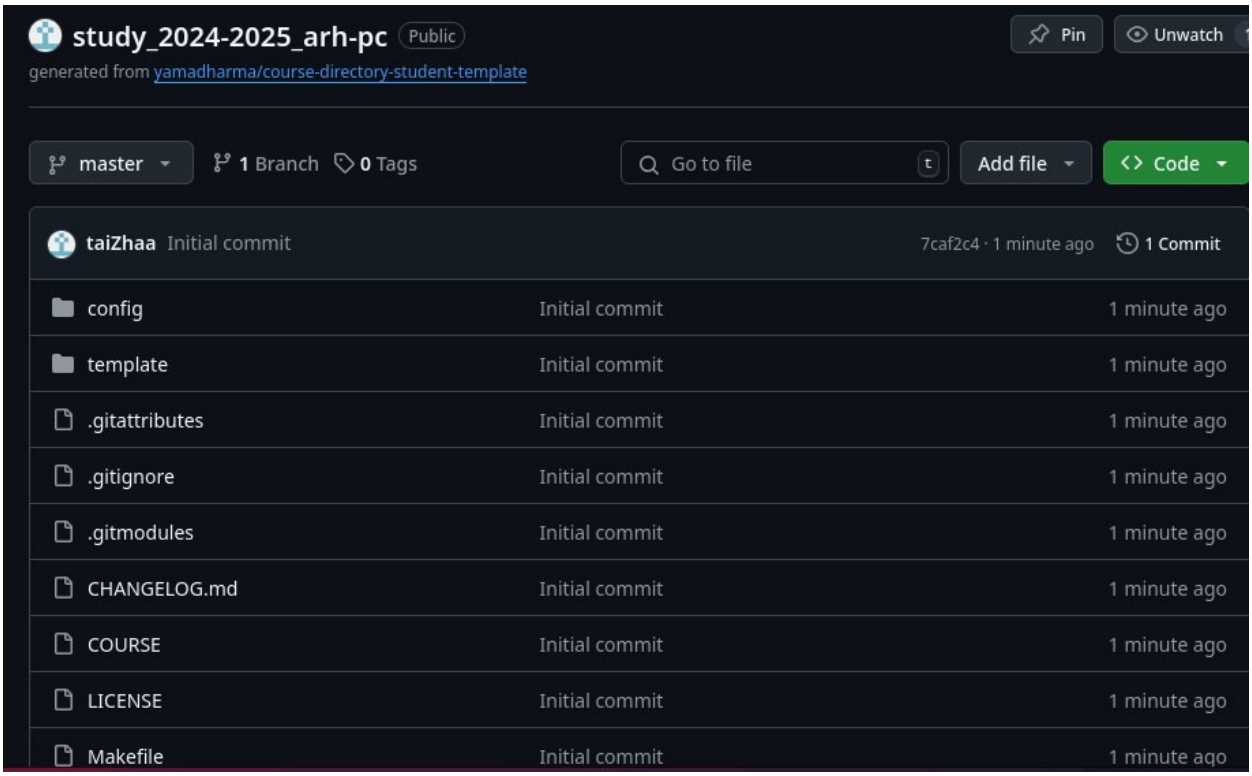


Рис. 4.16: Созданный репозиторий

Через терминал перехожу в созданный каталог курса с помощью утилиты cd

(рис. 4.17).

```
taizha@fedora:~$ cd ~/work/study/2024-2025/Архитектура\ компьютера/
```

Рис. 4.17: Перемещение между директориями

Клонирую созданный репозиторий с помощью команды git clone –recursive git@github.com:/study_2024–2025_arh_pc.git (рис. 4.18).

```
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера$ git clone --recursive git@gi
thub.com:taiZhaa/study_2024-2025_arh-pc.git
Клонирование в «study_2024-2025_arh-pc»...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 33 (delta 1), reused 18 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (33/33), 18.82 КиБ | 4.70 МиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-ma
rkdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-tem
plate.git) зарегистрирован по пути «template/report»
Клонирование в «/home/taizha/work/study/2024-2025/Архитектура компьютера/study_2024-2025_
arh-pc/template/presentation»...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
```

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).

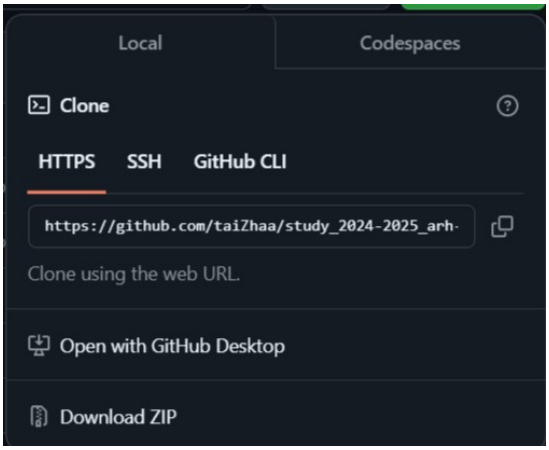


Рис. 4.19: Окно с ссылкой для копирования репозитория

4.6 Настройка каталога курса

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

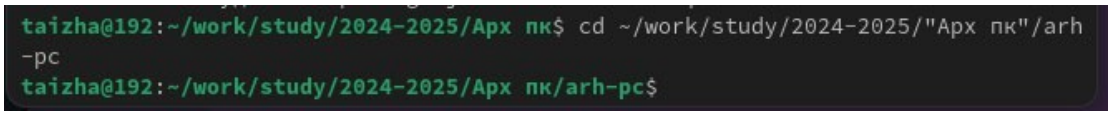


Рис. 4.20: Перемещение между директориями

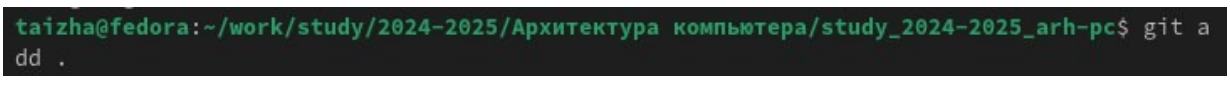
Удаляю лишние файлы с помощью утилиты rm (рис. 4.21).

Рис. 4.21: Удаление файлов

Создаю необходимые каталоги (рис. 4.22).

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).



```
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ git c
ommit -am 'feat(main): make course structure'
[master 0719129] feat(main): make course structure
 223 files changed, 53681 insertions(+), 14 deletions(-)
 create mode 100644 labs/README.md
 create mode 100644 labs/README.ru.md
 create mode 100644 labs/lab01/presentation/.projectile
 create mode 100644 labs/lab01/presentation/.texlabroot
 create mode 100644 labs/lab01/presentation/Makefile
 create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
 create mode 100644 labs/lab01/presentation/presentation.md
 create mode 100644 labs/lab01/report/Makefile
 create mode 100644 labs/lab01/report/bib/cite.bib
 create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
 create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
 create mode 100755 labs/lab01/report/pandoc/filters/pandoc_eqnos.py
 create mode 100755 labs/lab01/report/pandoc/filters/pandoc_fignos.py
 create mode 100755 labs/lab01/report/pandoc/filters/pandoc_secnos.py
 create mode 100755 labs/lab01/report/pandoc/filters/pandoc_tablenos.py
 create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/__init__.py
 create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/core.py
```

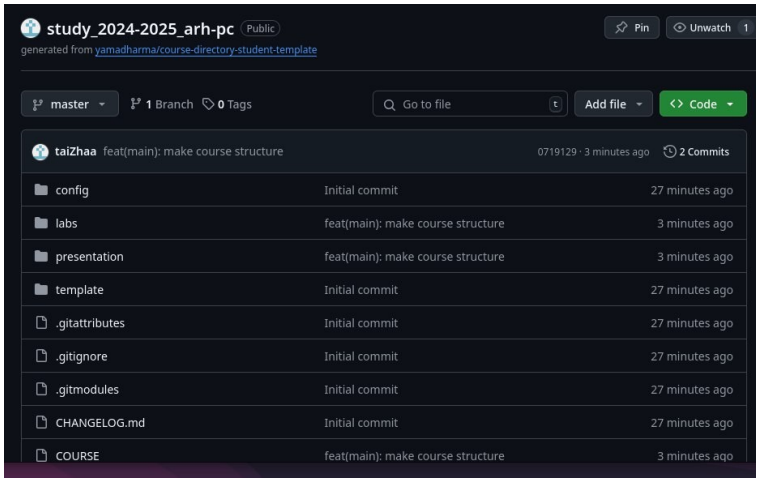
Рис. 4.23: Добавление и сохранение изменений на сервере

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

```
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ git p
ush
Перечисление объектов: 37, готово.
Подсчет объектов: 100% (37/37), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (29/29), готово.
Запись объектов: 100% (35/35), 341.40 КиБ | 2.84 МиБ/с, готово.
Total 35 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:taiZhaa/study_2024-2025_arh-pc.git
   7caf2c4..0719129  master -> master
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$
```

Рис. 4.24: Выгрузка изменений на сервер

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



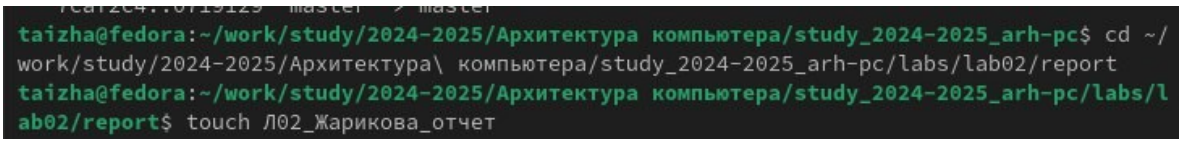


Рис. 4.25: Страница репозитория

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 третьей лабораторной работе с помощью утилиты touch (рис. 4.26).

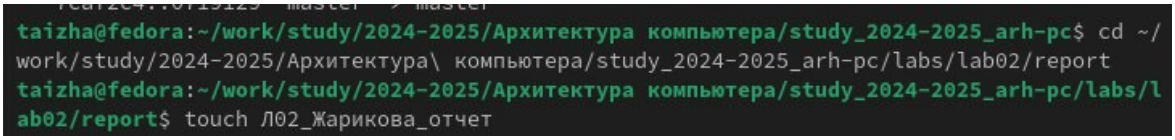


Рис. 4.26: Создание файла

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

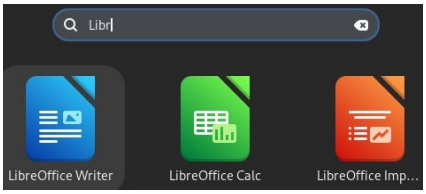


Рис. 4.27: Меню приложений

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

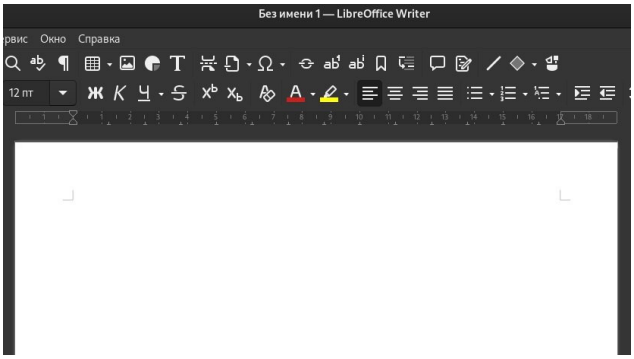


Рис. 4.28: Работа с отчетом в текстовом процессоре

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).

-

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).

-

Рис. 4.31: Копирование файла

Перехожу из подкаталога lab01/report в подкаталог lab02/report с помощью утилиты cd (рис. 4.32).

-

Рис. 4.32: Перемещение между директориями

Копирую вторую лабораторную с помощью утилиты cp и проверяю правильность выполнения команды cp с помощью ls (рис. 4.33).

-

Рис. 4.33: Копирование файла

3. Добавляю с помощью команды git add в коммит созданные файлы: Л02_Жарикова_отчет (рис. 4.34).

-

Рис. 4.34: Добавление файла на сервер

Перехожу в директорию, в которой находится отчет по первой лабораторной работе с помощью cd (рис. 4.35).

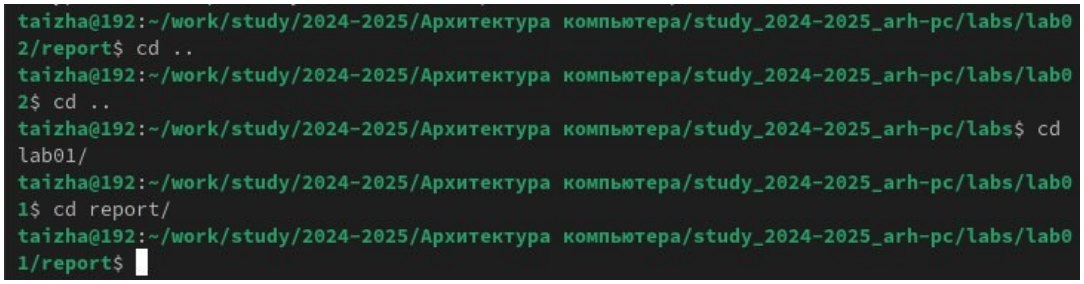


Рис. 4.35: Перемещение между директориями

Добавляю файл Л01_Жарикова_отчет (рис. 4.36).

-

Рис. 4.36: Добавление файла на сервер

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

-

Рис. 4.37: Подкаталоги и файлы в репозитории

Отправляю в центральный репозиторий сохраненные изменения командой git push -f origin master (рис. 4.38).

-

Рис. 4.38: Отправка в центральный репозиторий сохраненных изменений

Проверяю на сайте GitHub правильность выполнения заданий. Вижу, что пояснение к совершенным действиям отображается (рис. 4.39).

-

Рис. 4.39: Страница каталога в репозитории

При просмотре изменений так же вижу, что были добавлены файлы с отчетами по лабораторным работам (рис. 4.40).

-

Рис. 4.40: Страница последних изменений в репозитории

Вижу, что отчеты по лабораторным работам находятся в соответствующих каталогах репозитория: отчет по первой - в lab01/report (рис. 4.41), по второй – в lab02/report (рис. 4.42).

-

Рис. 4.41: Каталог lab01/report

-

Рис. 4.42: 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