

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: Первий Анастасия Андреевна

Группа: НПИбд-03-23

МОСКВА

2023_г.

Содержание

1. Список иллюстраций	3
2. Цель работы	4
3. Теоретическое введение	5
4. Задание	6
5. Выполнение лабораторной работы	7
6. Вывод.....	15

1. Список иллюстраций

Рис. 1 Учетная запись на сайте github	7
Рис. 2 Выполнение команд <code>git config -- global user.name "<NameSurname>"</code> и <code>git config -- global user.email "<work@mail>"</code>	7
Рис. 3 Настройка utf-8 в выводе сообщений.....	8
Рис. 4 Присваивание параметров и имени для начальной ветки.....	8
Рис. 5 Генерация SSH-ключа.....	8
Рис. 6 Копирование ключа.....	9
Рис. 7 Добавление ключа	9
Рис. 8 Создание рабочего пространства.....	9
Рис. 9 Проверка создания рабочего пространства	10
Рис. 10 Шаблон репозитория.....	10
Рис. 11 Создание репозитория	10
Рис. 12 Клонирования репозитория.....	11
Рис. 13 Окно с ссылкой для клонирования репозитория	11
Рис. 14 Переход в каталог arch-pc.....	12
Рис. 15 Удаление лишних файлов.....	12
Рис. 16 Создание каталога Course.....	12
Рис. 17 Обновление данных в репозитории.....	12
Рис. 18 Обновление данных в репозитории.....	12
Рис. 19 Страница репозитория	13
Рис. 20 Перемещение отчета по первой лабораторной работе	13
Рис. 21 Проверка каталога lab01	14

2. Цель работы

Изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

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 неполную 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. Задание

1. Настройка github
2. Базовая настройка git
3. Создание SSH ключа
4. Создание рабочего пространства
5. Создание репозитория курса на основе шаблона
6. Настройка каталога курса
7. Задание для самостоятельной работы

5. Выполнение лабораторной работы

Настройка github

Ранее я уже создала учетную запись и заполнила основные данные на сате <https://github.com/>, поэтому сейчас я не создала новую, а буду использовать уже существующую

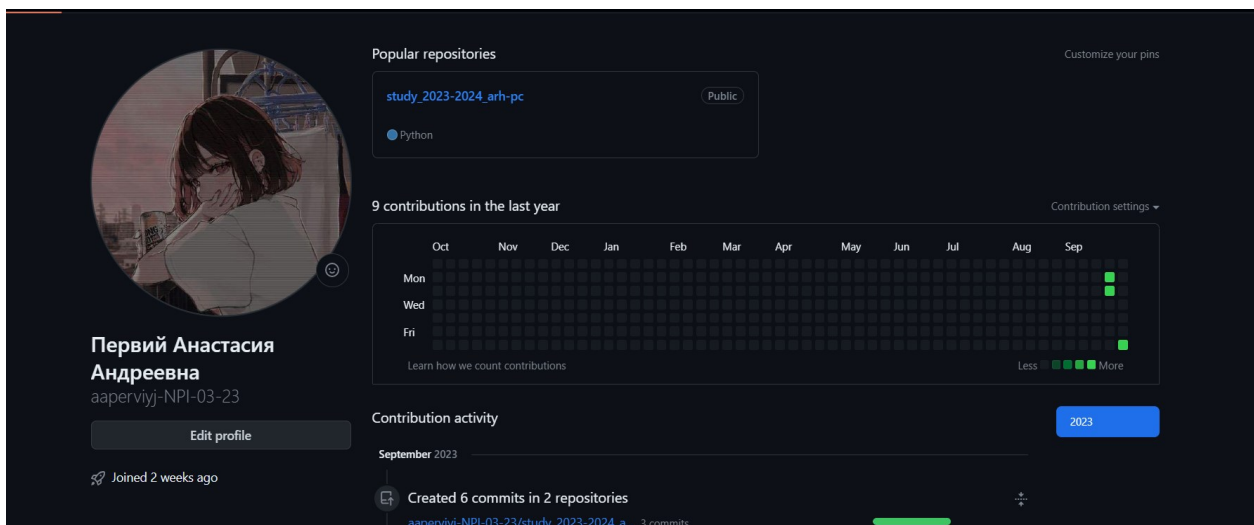


Рис. 1 Учетная запись на сайте github

Базовая настройка git

Открываю терминал в виртуальной машине и ввожу следующие команды, указав имя и email владельца репозитория(меня):

```
git config -- global user.name "aapervij-NPI-03-23"
```

```
git config -- global user.email anastasiapervij@gmail.com
```

```
aapervijj@aapervijj--NPI-03-23:~$ git config --global user.name "aapervij-NPI-03-23"
aapervijj@aapervijj--NPI-03-23:~$ git config --global user.email "anastasiapervij83@gmail.com"
aapervijj@aapervijj--NPI-03-23:~$
```

Рис. 2 Выполнение команд `git config -- global user.name "<NameSurname>"` и `git config -- global user.email "<work@mail>"`

Теперь нужно настроить utf-8 в выводе сообщений git, чтобы символы корректно отображались

```
aaperviyj@aaperviyj--NPI-03-23:~$ git config --global core.quotepath false
aaperviyj@aaperviyj--NPI-03-23:~$
```

Рис. 3 Настройка utf-8 в выводе сообщений

Задаю имя начальной ветке, а также параметры **autocrlf** и **safecrlf**, причем параметр **autocrlf** дополняем значением **input**, для конвертации символов разрыва строки в текстовых файлах (CRLF и LF) только при коммитах. Параметру **safecrlf** задаю значение **warn**, так Git будет проверять преобразование на обратимость, и при данном значении будет выведено только предупреждение, а необратимые конвертации будут приняты

```
aaperviyj@aaperviyj--NPI-03-23:~$ git config --global init.defaultBranch master
aaperviyj@aaperviyj--NPI-03-23:~$ git config --global core.autocrlf input
aaperviyj@aaperviyj--NPI-03-23:~$ git config --global core.safecrlf warn
aaperviyj@aaperviyj--NPI-03-23:~$
```

Рис. 4 Присваивание параметров и имени для начальной ветки

Создание SSH ключа.

На данном этапе необходимо сгенерировать ключи для последующей идентификации пользователя. Для этого я ввела команду **ssh-keygen -C "Имя Фамилия <work@email>"**

```
aaperviyj@aaperviyj--NPI-03-23:~$ ssh-keygen -C "Первый Анастасия Андреевна anastasiapervij83@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aaperviyj/.ssh/id_rsa):
/home/aaperviyj/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aaperviyj/.ssh/id_rsa
Your public key has been saved in /home/aaperviyj/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:DEeW35Amss2oYI/LIVzdLSFJa5Ss0t/o4xsQ7e890N0 Первый Анастасия Андреевна anastasiapervij83@gmail.com
The key's randomart image is:
+---[RSA 3072]-----+
|  ooo o. . |
| o=o+o + |
| ..O++*++ o |
| .o++o*o.. . |
| ..o=.oo S |
| ..o +o.. |
| o o.. .o . |
| o ooo.o E |
| .oo.... |
+---[SHA256]-----+
```

Рис. 5 Генерация SSH-ключа

Утилита **xclip** уже была установлена ранее, до выполнения данной лабораторной работы, с помощью команды **sudo apt install xclip**. Поэтому я пропускаю ее установку и перехожу сразу к этапу копирования открытого ключа из директории, в которую автоматически сохранился ключ через команду **cat ~/.ssh/id_rsa.pub | xclip**


```
aaaperviyj@aaaperviyj--NPI-03-23:~$ cat ~/.ssh/id_rsa.pub | xclip
```

Рис. 6 Копирование ключа

Открываю браузер и захожу на сайт [SSH and GPG keys \(github.com\)](https://ssh.github.com). Нажимаю кнопку “New SSH key” Вставляю ключ в поле Key и задаю имя для ключа. Затем нажимаю кнопку Add SSH-key чтобы завершить добавление ключа

Add new SSH Key

Title

Key type

Authentication Key ↕

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGC38hgefz09lQ+pphUZRxnbLtlCimmGCa9dK+91ENdljLrlxnmGdYYSL3U0G
RluH5GO3ImT7NZ00rMXFKDy52AD1KT7zitoTj/8F
/ibz8sN+1mO3o1ZzZBmeIOcW5Zu0XpgB80qcNwllVAaDPGPJv2bnDZtcq2R0
/Bf4HrngWZp80Eo2D5C1HwEQ8Ym4ljWZFdmvMb3HQcltU0EFwAgSpJqnvJuiGrdGP4mbHtCB3t2dFor8crAnYcjxblHd
/N/U76rcsxVnK+0bia0HyfC5Xy4LPvpUo+ZhXQLGh8dfw3ZzfAL+whhcvoOLlyE8JVIIH25tV3OEuaLQtQ3jbaADxuG5vB
Fe4pYNf8/nzrsS
/CydgPrkKfcRld+Zs+cA12Pt9YMJSHMpu4U042JPf2W5wASx0aaND8YE7Dg3gVEM7zEYADyyjV2q5rL6+BcqzBaFWjSj
Bystoa4dxad8o9QPE7/1vdWRs188U2c4+svyxndGJ3AgnuMleBfU4ZjpoeU3ds= Первый Анастасия Андреевна
anastasiapervij83@gmail.com
```

Add SSH key

Рис. 7 Добавление ключа

Создание рабочего пространства

Открываю терминал в виртуальной машине, чтобы создать рабочее пространство с помощью команды `mkdir -p ~/work/study/2023-2024/"Архитектура компьютера"` и далее проверяю создан ли он с помощью команды `ls`

```
aaaperviyj@aaaperviyj--NPI-03-23:~$ mkdir -p ~/work/study/2023-2024/"Архитектура компьютера"
```

Рис. 8 Создание рабочего пространства

```
aaaperviyj@aaaperviyj--NPI-03-23:~$ ls
snap  tmp  Видео  Загрузки  Музыка  'Рабочий стол'
temp  work  Документы  Изображения  Общедоступные  Шаблоны
aaaperviyj@aaaperviyj--NPI-03-23:~$
```

Рис. 9 Проверка создания рабочего пространства

Создание репозитория курса на основе шаблона

Сначала я перешла на страницу репозитория с шаблоном курса и использовала этот шаблон, создав новый репозиторий, на основе этого шаблона по ссылке [yamadharm/course-directory-student-template: Course Catalog Template for Students \(github.com\)](https://github.com/yamadharma/course-directory-student-template) Далее нажимаю Use this template и Create a new repository чтобы использовать этот шаблон для создания своего репозитория

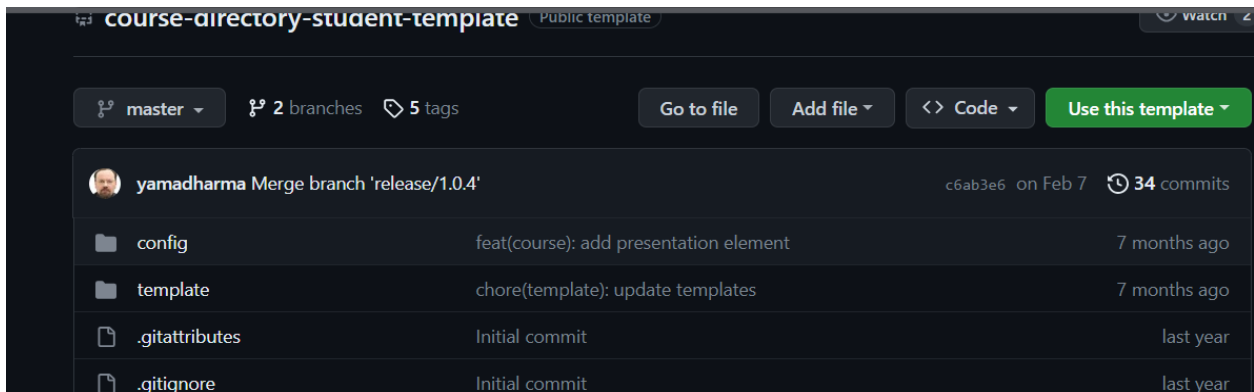


Рис. 10 Шаблон репозитория

В открывшемся окне нужно задать имя репозитория study_2023-2024_arh-pc

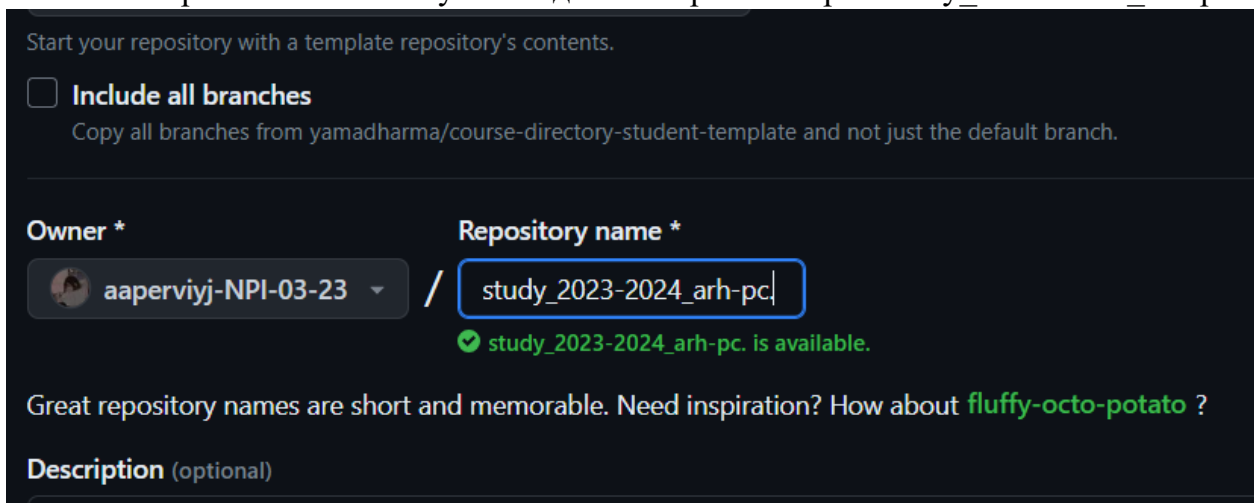


Рис. 11 Создание репозитория

Снова открываю терминал в виртуальной машине и клонирую созданный репозиторий с помощью команды `git clone --recursive git@github.com:study_2023-2024_arh-pc.git arch-pc`

```
aaperviyj@aaperviyj--NPI-03-23:~/work/study/2023-2024/Архитектура компьютера$ git clone --recursive git@github.com:aaperviyj-NPI-03-23/study_2023-2024_arh-pc.git arch-pc
Клонирование в «arch-pc»...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
Получение объектов: 100% (27/27), 17.00 КиБ | 3.40 МиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/aaperviyj/work/study/2023-2024/Архитектура компьютера/arch-pc/template/prese
```

Рис. 12 Клонирования репозитория

Ссылку для копирования можно взять на странице созданного репозитория. Для этого нужно нажать на **Code** и выбрать **SSH**

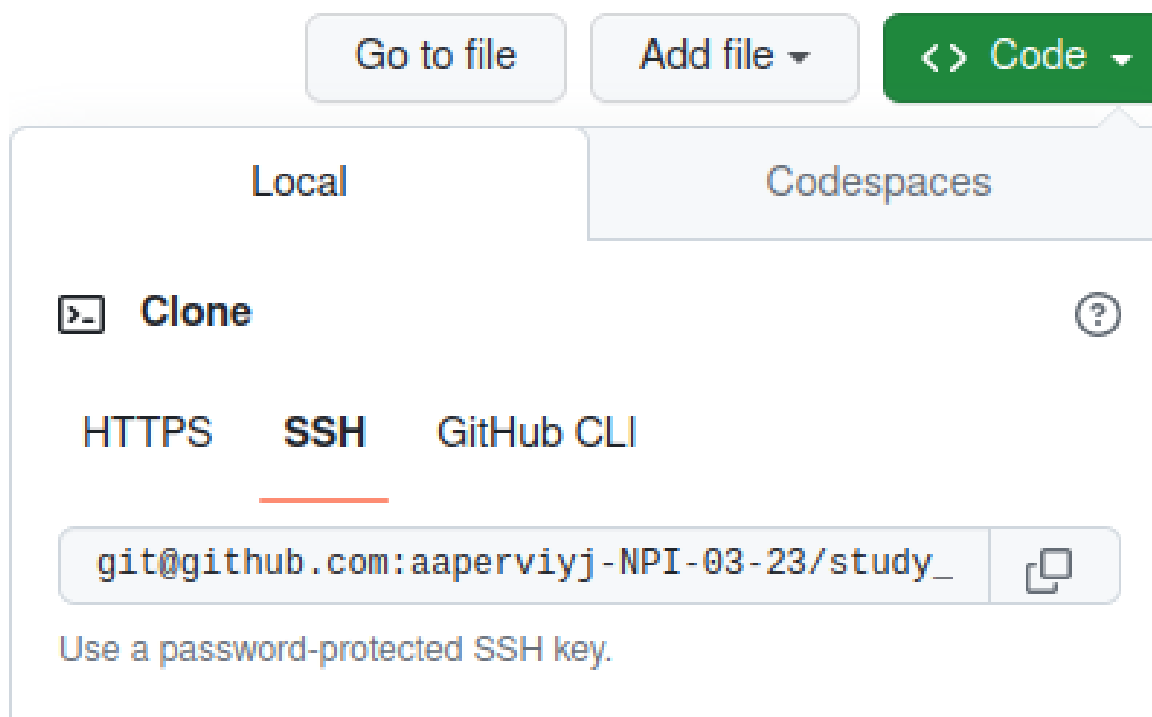


Рис. 13 Окно с ссылкой для клонирования репозитория

Настройка каталога курса

Перехожу в каталог `arch-pc` (Рис. 14). Затем удаляю лишние файлы, с помощью утилиты `rm` и создаю необходимые каталоги (Рис.15)

```
aaperviyj@aaperviyj--NPI-03-23:~/work/study/2023-2024/Архитектура компьютера$ cd arch-pc
aaperviyj@aaperviyj--NPI-03-23:~/work/study/2023-2024/Архитектура компьютера/arch-pc$
```

Рис. 14 Переход в каталог arch-pc

```
aaperviyj@aaperviyj--NPI-03-23:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ rm package.json
```

Рис. 15 Удаление лишних файлов

Теперь нужно создать каталог Course и отправить обновленный каталог в репозиторий

```
aaperviyj@aaperviyj--NPI-03-23:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE
aaperviyj@aaperviyj--NPI-03-23:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ make
```

Рис. 16 Создание каталога Course

Отправляю и сохраняю изменения с локального репозитория на сервер, с помощью команд:

git add

git commit -am 'feat(main); make course structure'

git push

```
aaperviyj@aaperviyj--NPI-03-23:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git add .
aaperviyj@aaperviyj--NPI-03-23:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git commit -am 'feat(main): make course structure'
[master 93d1512] feat(main): make course structure
199 files changed, 54725 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/Makefile
```

Рис. 17 Обновление данных в репозитории

```
aaperviyj@aaperviyj--NPI-03-23:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 37, готово.
Подсчет объектов: 100% (37/37), готово.
При сжатии изменений используется до 7 потоков
```

Рис. 18 Обновление данных в репозитории

Проверяю правильность выполнения всей работы на сайте aaperviyj-NPI-03-23/study_2023-2024_arh-pc (github.com) (Рис. 19)

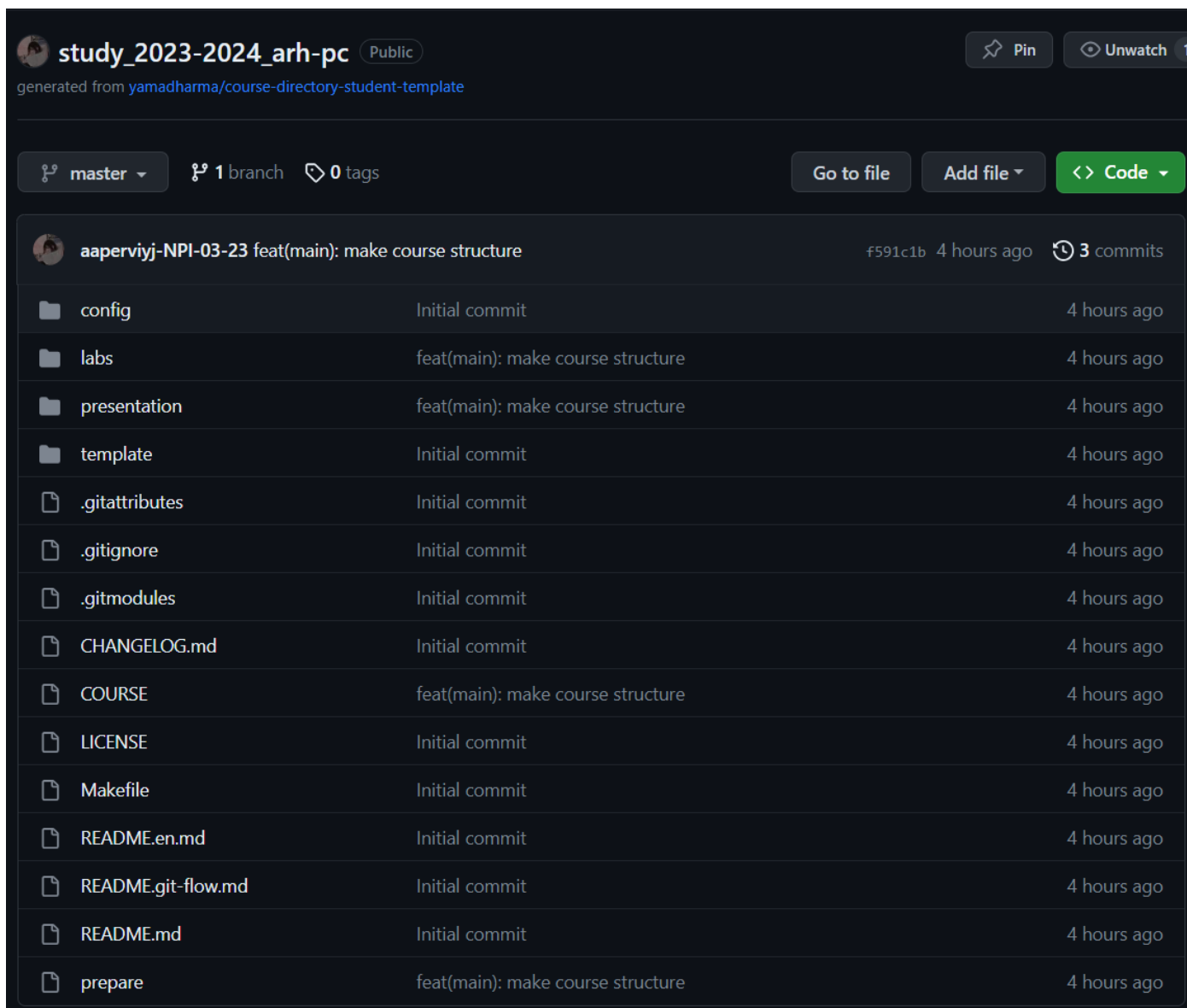


Рис. 19 Страница репозитория

Задание для самостоятельной работы

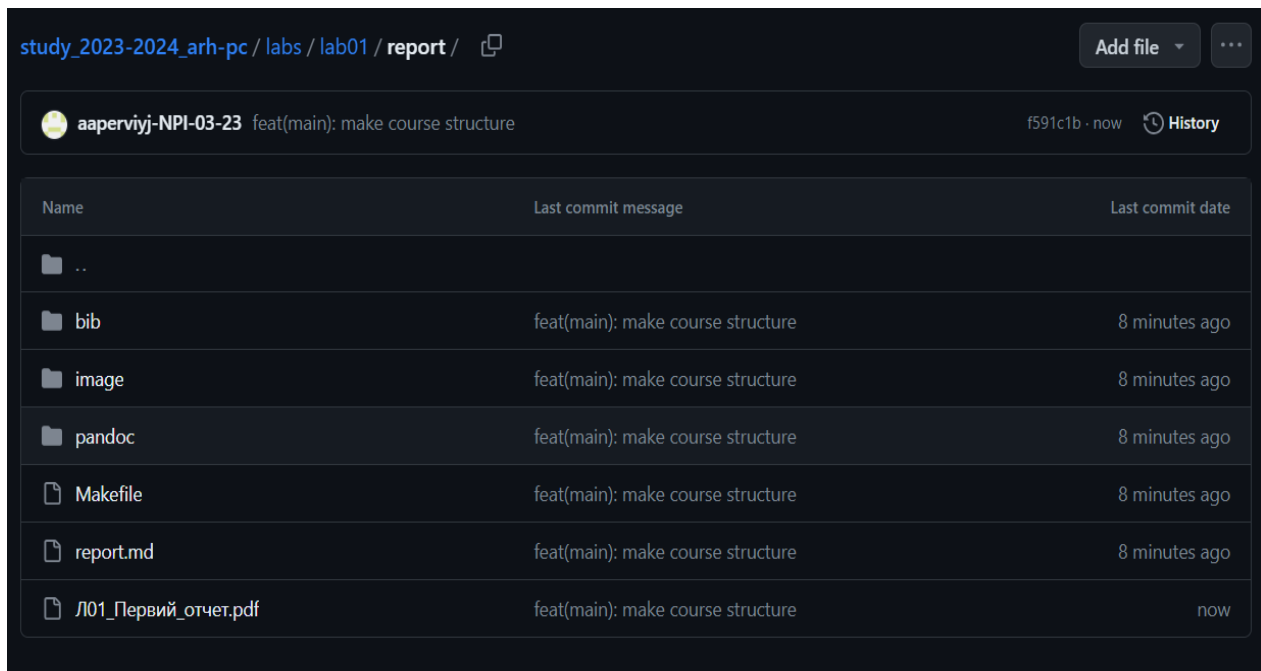
Открываю терминал в виртуальной машине и отчет, по первой лабораторной работе, отправляю в каталог report, с помощью команды **mv** через путь: **~work/.../lab01/report**

Далее нужно будет обновить данные в репозитории в github

```
aaperviyj@aaperviyj--NPI-03-23:~$ mv 'Л01_Первый отчет.pdf' ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report
aaperviyj@aaperviyj--NPI-03-23:~$ cd ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report
aaperviyj@aaperviyj--NPI-03-23:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report$ ls
bib image Makefile pandoc report.md Л01_Первый отчет.pdf
aaperviyj@aaperviyj--NPI-03-23:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report$ git add .
```

Рис. 20 Перемещение отчета по первой лабораторной работе

Проверяем правильность выполнения команд на странице репозитория



The screenshot shows a GitHub repository page for the path `study_2023-2024_arh-pc / labs / lab01 / report /`. The commit is by user `aaperviyj-NPI-03-23` with the message `feat(main): make course structure` and hash `f591c1b`. The commit date is `now`. Below the commit information is a table listing the files and folders in the repository.

Name	Last commit message	Last commit date
..		
bib	feat(main): make course structure	8 minutes ago
image	feat(main): make course structure	8 minutes ago
pandoc	feat(main): make course structure	8 minutes ago
Makefile	feat(main): make course structure	8 minutes ago
report.md	feat(main): make course structure	8 minutes ago
Л01_Первый_отчет.pdf	feat(main): make course structure	now

Рис. 21 Проверка каталога lab01

Так как отчет по второй лабораторной работе я делаю на момент выполнения данного пункта, я **не могу** отправить его на github, поэтому отправляют пока отчет только по первой лабораторной работе.

6. Вывод

Я приобрела практический опыт работы с системой git, изучила принципы и применение контроля версий.