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: Ниамби Давид Бени

Группа:НБИбд-02-25

МОСКВА

2025 г.

1. Цель работы :

Целью работы является изучение идеологии и применения средств контроля версий, приобретение практических навыков по работе с системой контроля версий git.

2. Теоретическое введение

2.1. Системы контроля версий. Общие понятия

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 Демидова А. В., Велиева Т. Р., Геворкян М. Н. 19 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.

2.2. Система контроля версий Git

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.

2.3. Основные команды git

git init- создание основного дерева репозитория

git pull- получение обновлений (изменений) текущего дерева из центрального репозитория

git push -отправка всех произведённых изменений локального дерева в центральный репозиторий

git status- просмотр списка изменённых файлов в текущей директории

git diff- просмотр текущих изменений git add . добавить все изменённые и/или созданные файлы и/или каталоги

git add

имена_файлов - добавить конкретные изменённые и/или созданные файлы и/или каталоги

git rm

имена_файлов - удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)

git commit -am

'Описание коммита' - сохранить все добавленные изменения и все изменённые файлы

git checkout -b

имя_ветки - создание новой ветки, базирующейся на текущей git checkout имя_ветки переключение на некоторую ветку (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)

git push origin

имя_ветки - отправка изменений конкретной ветки в центральный репозиторий git merge --no-ff имя_ветки слияние ветки с текущим деревом

git branch -d

имя_ветки - удаление локальной уже слитой с основным деревом ветки

git branch -D

имя_ветки - принудительное удаление локальной ветки

git push origin

:имя_ветки - удаление ветки с центрального репозитория

2.4. Стандартные процедуры работы при наличии центрального репозитория

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):

```
git checkout master
git pull
git checkout -b имя_ветки
```

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. Для этого необходимо проверить, какие файлы изменились к текущему моменту:

git status

При необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий. Затем полезно просмотреть текст изменений на предмет соответствия правилам ведения чистых коммитов:

git diff

Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых нужно сохранить. Для этого используем команды добавления и/или удаления с нужными опциями:

git add имена_файлов
git rm имена_файлов

Если нужно сохранить все изменения в текущем каталоге, то используем:

```
git add .
```

Затем сохраняем изменения, поясняя, что было сделано:

git commit -am "Some commit message"

и отправляем в центральный репозиторий:

```
git push origin имя_ветки
```

или

git push

3. выполнения лабораторной работы

3.1. Настройка github

Создайте учётную запись на сайте https://github.com/ и заполните основные данные.(рис1)

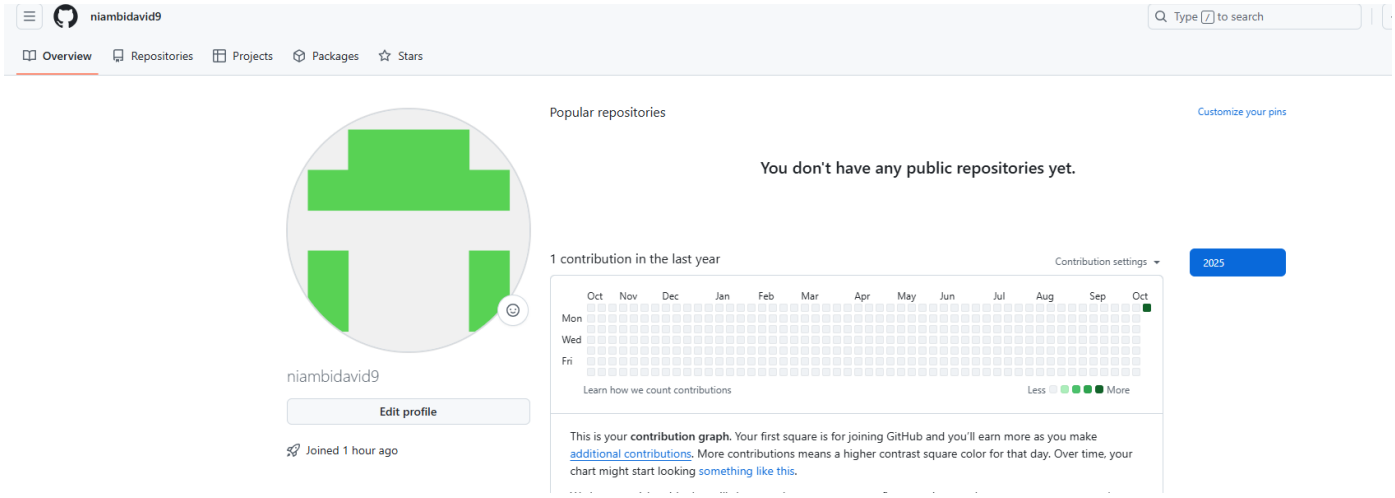


рис1 : Создание учётной записи на сайте Github

3.2 Базовая настройка git

Сначала сделаем предварительную конфигурацию git. Откройте терминал и введите следующие команды, указав имя и e-mail.(рис2)

```
davidbeniniambi@fedora:~$ git config --global user.name "<niambidavid9>"
davidbeniniambi@fedora:~$ git config --global user.email "<niambibeni77@gmail.com>"
```

Настроим utf-8 в выводе сообщений git:

```
davidbeniniambi@fedora:~$ git config --global user.name "<niambidavid9>"
davidbeniniambi@fedora:~$ git config --global user.email "<niambibeni77@gmail.com>"
davidbeniniambi@fedora:~$ git config --global core.quotepath false
davidbeniniambi@fedora:~$
```

Зададим имя начальной ветки (будем называть её master):

```
davidbeniniambi@fedora:~$ git config --global user.name "<niambidavid9>"
davidbeniniambi@fedora:~$ git config --global user.email "<niambibeni77@gmail.com>"
davidbeniniambi@fedora:~$ git config --global core.quotepath false
davidbeniniambi@fedora:~$ git config --global init.defaultbranch master
davidbeniniambi@fedora:~$
```

Параметр autocrlf:

```
davidbeniniambi@fedora:~$ git config --global user.name "<niambidavid9>"
davidbeniniambi@fedora:~$ git config --global user.email "<niambibeni77@gmail.com>"
davidbeniniambi@fedora:~$ git config --global core.quotepath false
davidbeniniambi@fedora:~$ git config --global init.defaultbranch master
davidbeniniambi@fedora:~$ git config --global core.autocrlf input
davidbeniniambi@fedora:~$
```

Параметр safecrlf:

```
davidbeniniambi@fedora:~$ git config --global user.name "<niambidavid9>"
davidbeniniambi@fedora:~$ git config --global user.email "<niambibeni77@gmail.com>"
davidbeniniambi@fedora:~$ git config --global core.quotepath false
davidbeniniambi@fedora:~$ git config --global init.defaultbranch master
davidbeniniambi@fedora:~$ git config --global core.autocrlf input
davidbeniniambi@fedora:~$ git config --global core.safecrlf warn
davidbeniniambi@fedora:~$
```

Создание SSH-ключа:

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый):

```
davidbeniniambi@fedora:~
davidbeniniambi@fedora:~$ ssh-keygen -C "David Niambi <niambibeni77@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/davidbeniniambi/.ssh/id_ed25519):
Created directory '/home/davidbeniniambi/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/davidbeniniambi/.ssh/id_ed25519
Your public key has been saved in /home/davidbeniniambi/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:kIorDzo9PT2SctIUFUZ3rlu0ixi332hQqgvlqNzlLQQ David Niambi <niambibeni77@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|     .+.. .      |
|     ..o o       |
|     .o   o      |
|   .E. . o..     |
|  . .oo Soo      |
|   ..+.+o= .     |
|o..+o++.+..      |
|o==oB+=. ..o     |
|..+=.oo+..o .    |
+----[SHA256]-----+
davidbeniniambi@fedora:~$
```

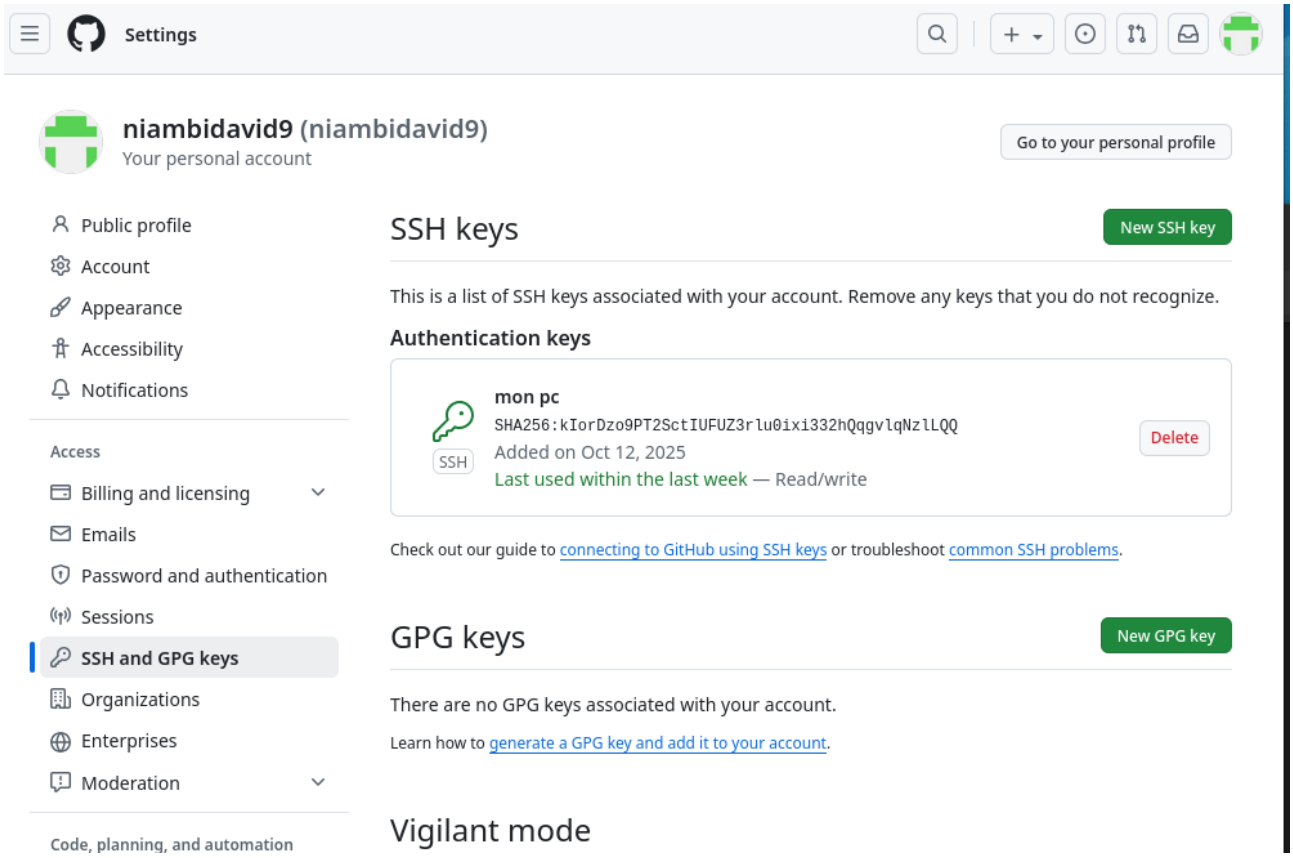
Ключи сохранятся в каталоге ~/.ssh/.

```
davidbeniniambi@fedora:~$ cd
davidbeniniambi@fedora:~$ cd ~/.ssh
davidbeniniambi@fedora:~/.ssh$ ls
id_ed25519  id_ed25519.pub
davidbeniniambi@fedora:~/.ssh$
```

Далее необходимо загрузить сгенерированный открытый ключ. Для этого следует 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 . Копируем из локальной консоли ключ в буфер обмена cat ~/.ssh/id_rsa.pub | xclip -sel clip

```
davidbeniniambi@fedora:~/.ssh$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIFSYrFFbBkQozmIee0PWm+OJ9P07AxequdibQcT0quy8 David Niambi <niambibeni77@gmail.com>
davidbeniniambi@fedora:~/.ssh$
```

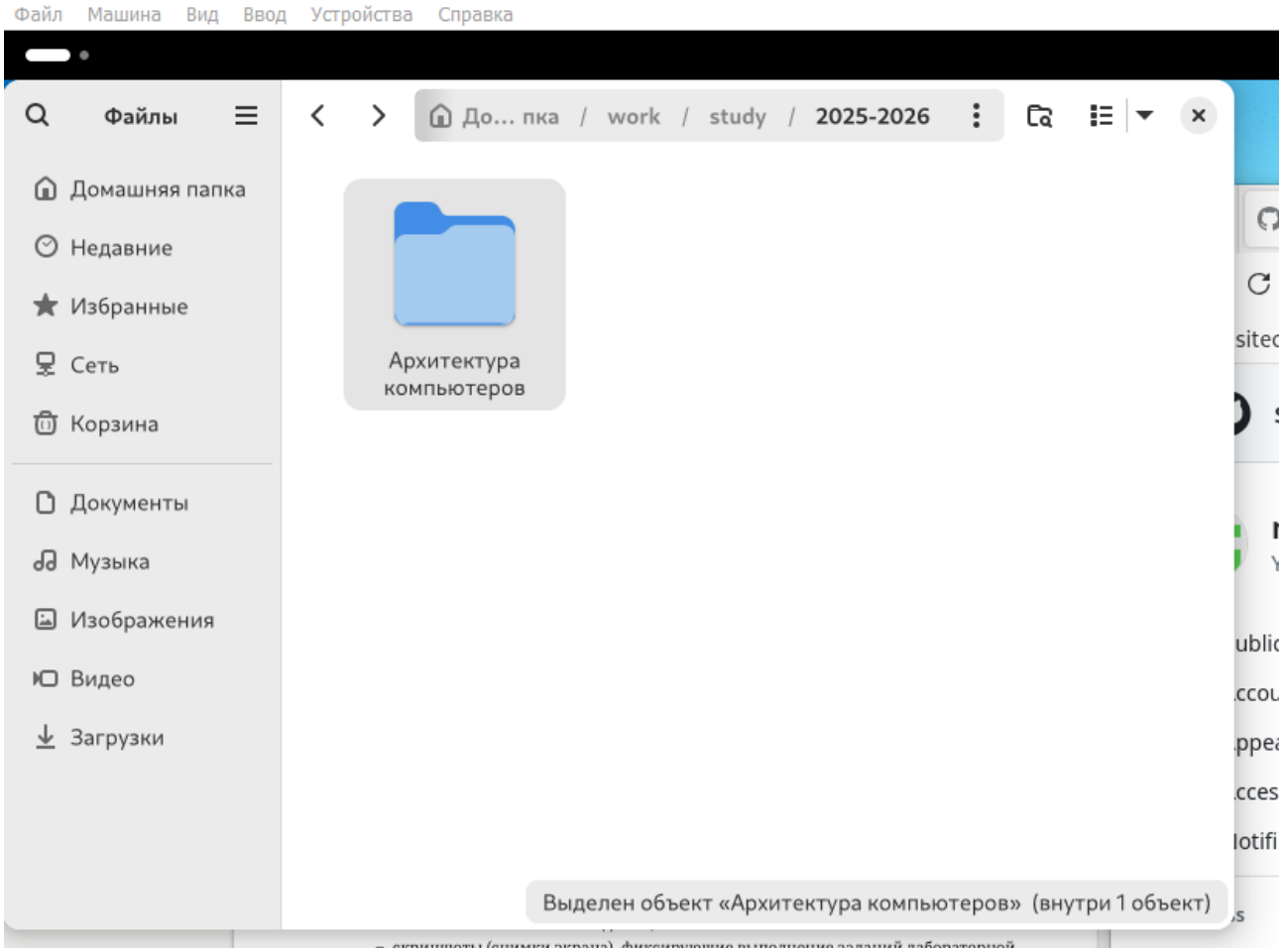
Вставляем ключ в появившееся на сайте поле и указываем для ключа имя (Title).



Создание рабочего пространства и репозитория курса на основе шаблона

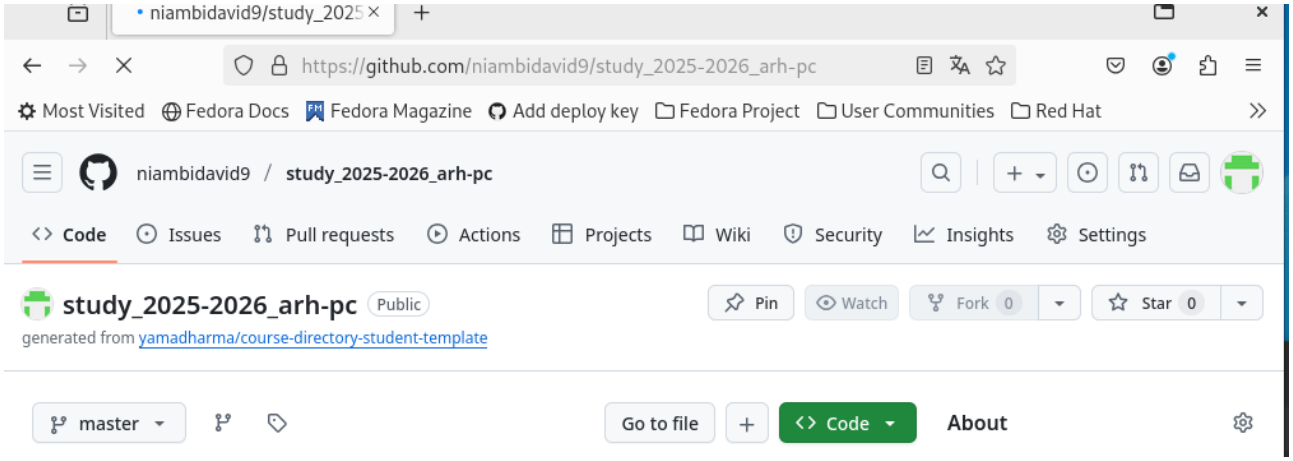
Название проекта на хостинге git имеет вид: study__ Например, для 2025–2026 учебного года и предмета «Архитектура компьютера» (код предмета arch-pc) название проекта примет следующий вид: study_2025–2026_arch-pc Откройте терминал и создайте каталог для предмета «Архитектура компьютера»:

```
mkdir -p ~/work/study/2023-2024/"Архитектура компьютера"
```



Создание репозитория курса на основе шаблона

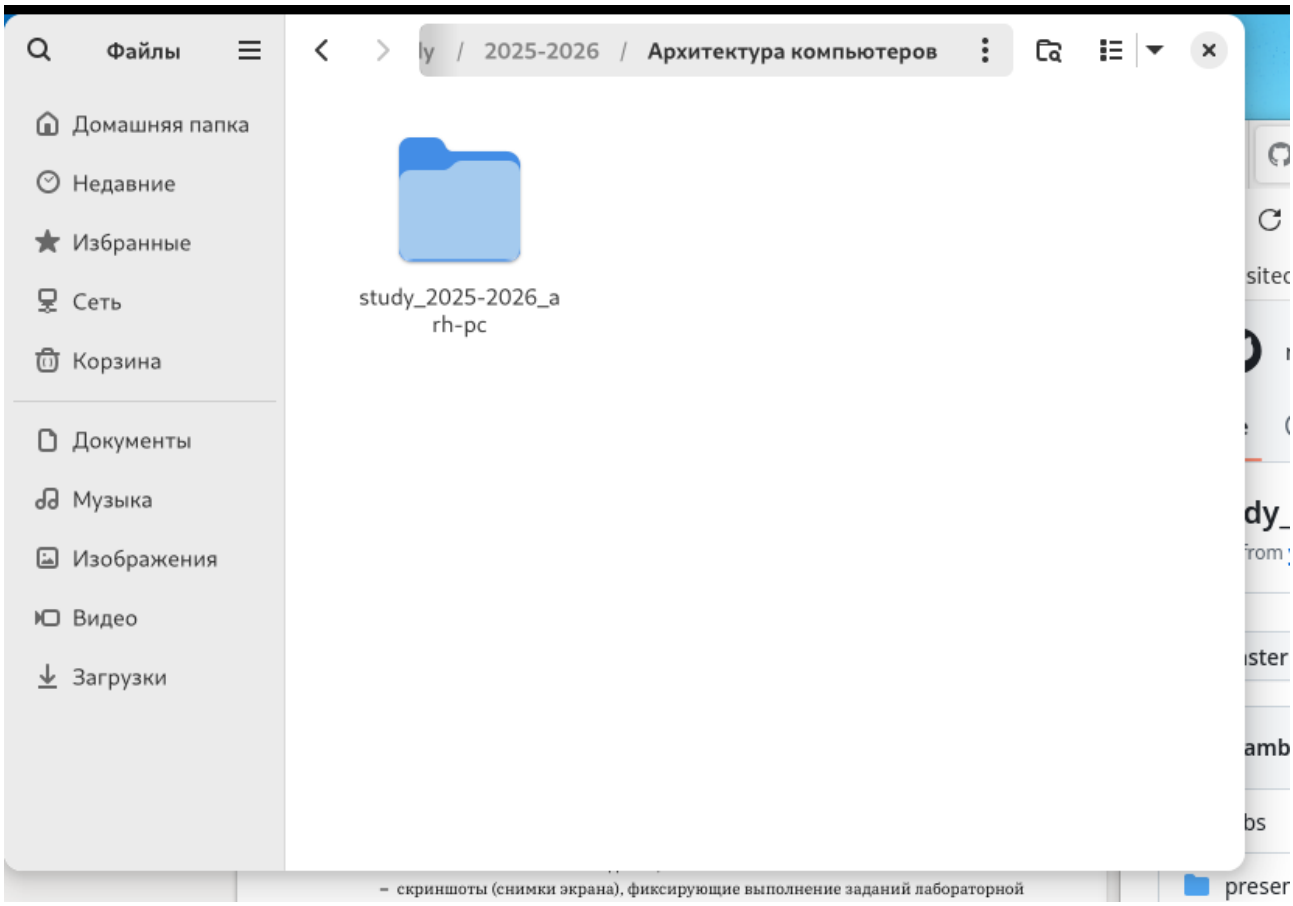
Репозиторий на основе шаблона можно создать через web-интерфейс github. Перейдите на станицу репозитория с шаблоном курса https://github.com/yamadharma/cour se-directory-student-template. Далее выберите Use this template (рис. ??). В открывшемся окне задайте имя репозитория (Repository name) study_2025–2026_arh-pc и создайте репозиторий (кнопка Create repository from template). Откройте терминал и перейдите в каталог курса:



cd ~/work/study/2025–2026/"Архитектура компьютера"

Клонируйте созданный репозиторий:

git clone --recursive git@github.com:/study_2025–2026_arh-pc.git arch-pc



Настройка каталога курса

Перейдите в каталог курса:

cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc

Создайте необходимые каталоги:

echo arch-pc > COURSE

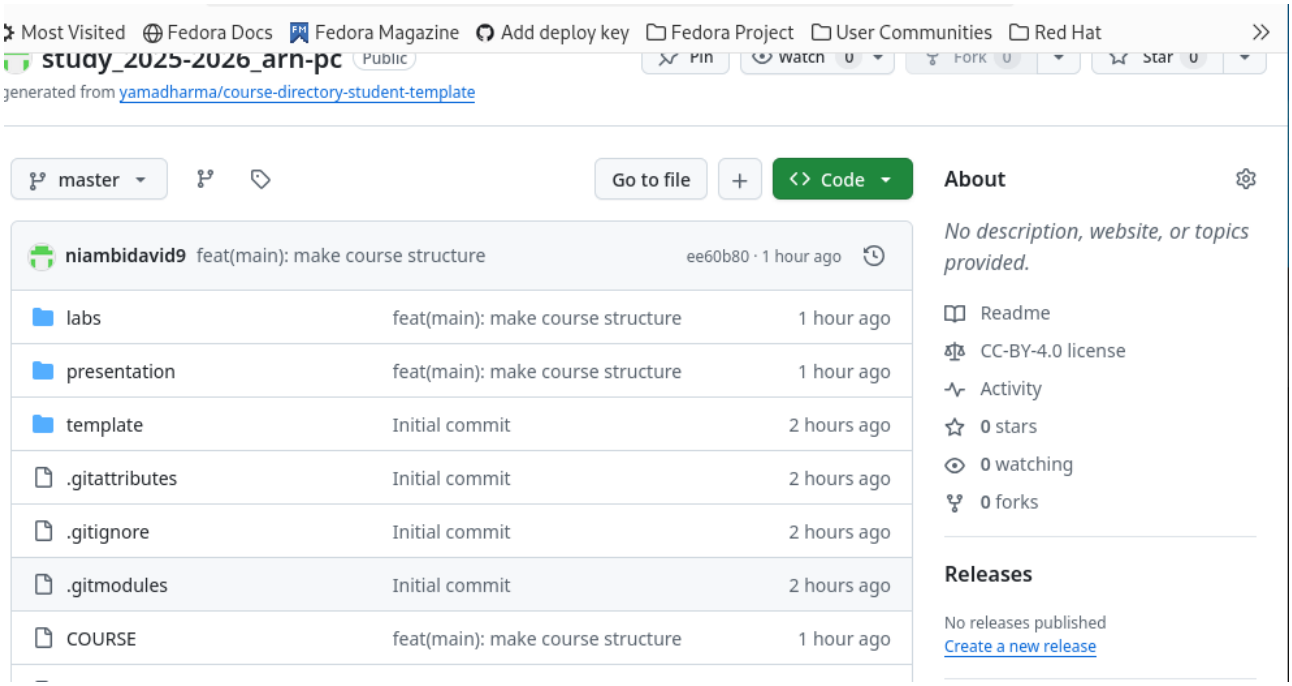
make prepare

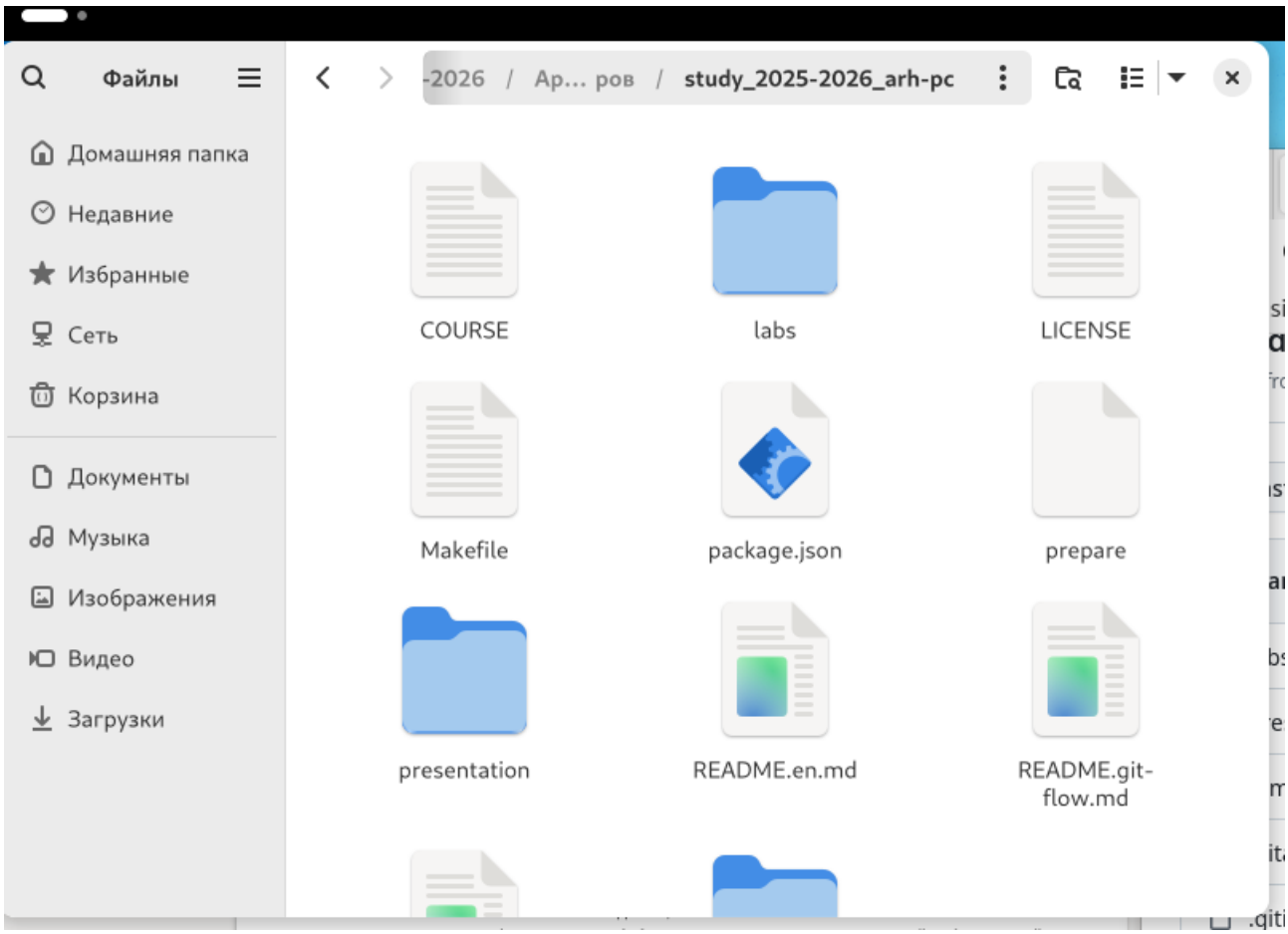
Отправьте файлы на сервер:

git add .

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

git push




Файлы
Домашняя папка
Недавние
Избранные
Сеть
Корзина
Документы
Музыка
Изображения
Видео
Загрузки
-2026 / Ар... ров / study_2025-2026_arh-pc
COURSE
labs
LICENSE
Makefile
package.json
prepare
presentation
README.en.md
README.git-flow.md