

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: Даровских Александра Сергеевна

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	21
Список литературы	22

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

Создаем учетную запись на сайте GitHub (рис. 4.1). Далее заполняем основные данные учетной записи.

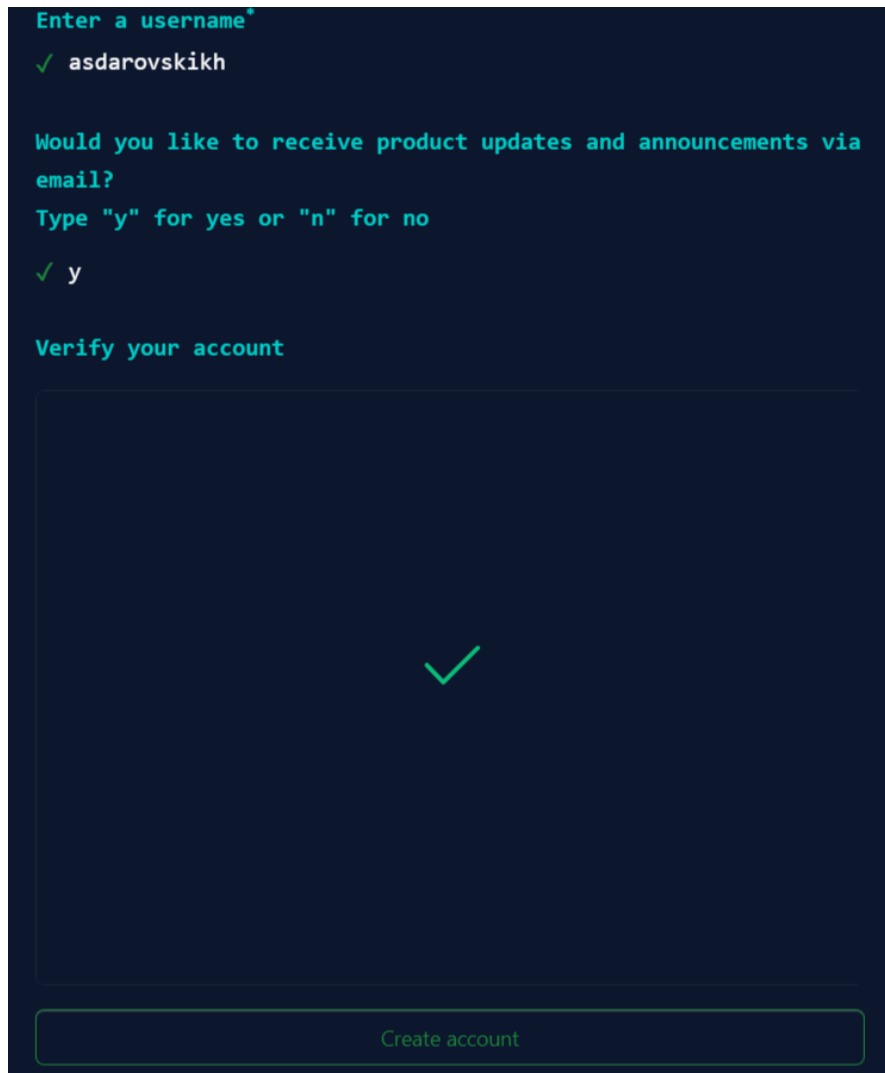
The image shows a dark-themed terminal window for creating a GitHub account. It contains the following text: 'Enter a username*' followed by '✓ asdarovskikh'; 'Would you like to receive product updates and announcements via email?' followed by 'Type "y" for yes or "n" for no' and '✓ y'; 'Verify your account' followed by a large green checkmark; and a 'Create account' button at the bottom.

Рис. 4.1. Заполнение данных учетной записи на GitHub

Подтверждение созданного аккаунта (рис. 4.2).

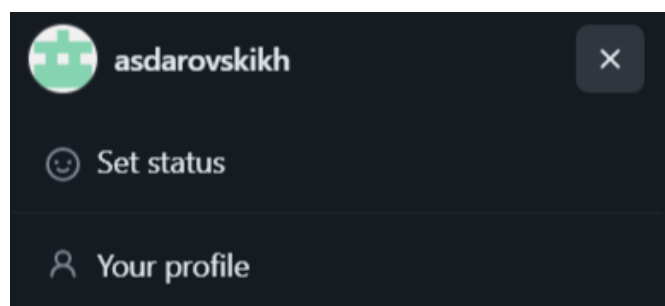


Рис. 4.2. Аккаунт GitHub

4.2 Базовая настройка Git

Открываем виртуальную машину, затем открываем терминал и делаем предварительную конфигурацию git. Вводим следующие команды `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 свою (рис. 4.3).

```
[asdarovskikh@fedora ~]$ git config --global user.name "<Aleksandra Darovskikh>"
[asdarovskikh@fedora ~]$ git config --global user.email "<1132232877@pfur.ru>"
```

Рис. 4.3 Предварительная конфигурация git

Настроим utf-8 в выводе сообщений git для корректного отображения символов . Зададим имя начальной ветки и параметры `autocrlf` и `safecrlf` (рис. 4.4).

```
[asdarovskikh@fedora ~]$ git config --global core.quotepath false
[asdarovskikh@fedora ~]$ git config --global init.defaultBranch master
[asdarovskikh@fedora ~]$ git config --global core.autocrlf input
[asdarovskikh@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 4.4 Настройка git

4.3. Создание SSH ключа

Генерируем приватный и открытый ключи для последующей идентификации пользователя на сервере (рис. 4.5.)

```
[asdarovskikh@fedora ~]$ ssh-keygen -C "Aleksandra Darovskikh <1132232877@pfur.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/asdarovskikh/.ssh/id_rsa):
Created directory '/home/asdarovskikh/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/asdarovskikh/.ssh/id_rsa
Your public key has been saved in /home/asdarovskikh/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Q35qu38cXg6HNiDZGddGD1Gq17Eb0LYtm2vgtS2pA5c Aleksandra Darovskikh <1132232877@pfur.ru>
The key's randomart image is:
+---[RSA 3072]-----+
|
|      o+o. |
|      . . ++ |
|    .o + o.+ |
|    oo + .o.= |
|    S....oo+o |
|    +. E.++o |
|    o  B Xo= |
|    . . * *.. |
|    oo...+.. |
+-----[SHA256]-----+
```

Рис. 4.5 Создание SSH ключа

Просмотреть ключ можно с помощью команды `cat`, далее скопировав и вставив его в поле для ключа на сайте (рис. 4.6.). После чего авторизуется ключ (рис. 4.7.).


```
[asdarovskikh@fedora ~]$ cat /home/asdarovskikh/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCs4I8bzPKsJT1tCbMMeLAdAsrJLxLS6cweLSKc/Gz6TZHk+kPsydbUSJPFxxSoZ6znf7IyUFm6T9D8yoeFUV2ir5PFv9PfsA3ZFnxZFUTYIFH
uPKoJldMPHGeiae/6wch27zSClg7BX6CkmWPJR2huOQDuZ9+$4UZNUCnqSj+EleNDbuyyWnI8oGW2ZjwPwycCfv0GOhzDK2VJxtW5o6LhgjEw3y19EDwvcq2g13p2eUHbd3PqfVXclp5nTW4qKd
TG/g188benNN7vU0P/Ed6AgOVKe08NQH1Kycr6PQmqOA9Y+NEyhR+UtceAHyH+ejeRg+ZZAjr4vnCHWp0zfvc07BPzNarQUJahmrEuUe/3b1Sfe44UKAx1A1XjYdZmyxHoXvUF45LJudKSA3VY
fNekyIBhVwcGJ47HAZ1z53RLnRKABgJDV2cyQfppfvVSuJoKxMnsJgFAvSb9wK0vNsSa9YzKes835hExA9ft+sm++10zvn80ftUwpJWqJU= Aleksandra Darovskikh <1132232877@pfur.
ru>
```

Рис. 4.6. Открытый ключ SSH

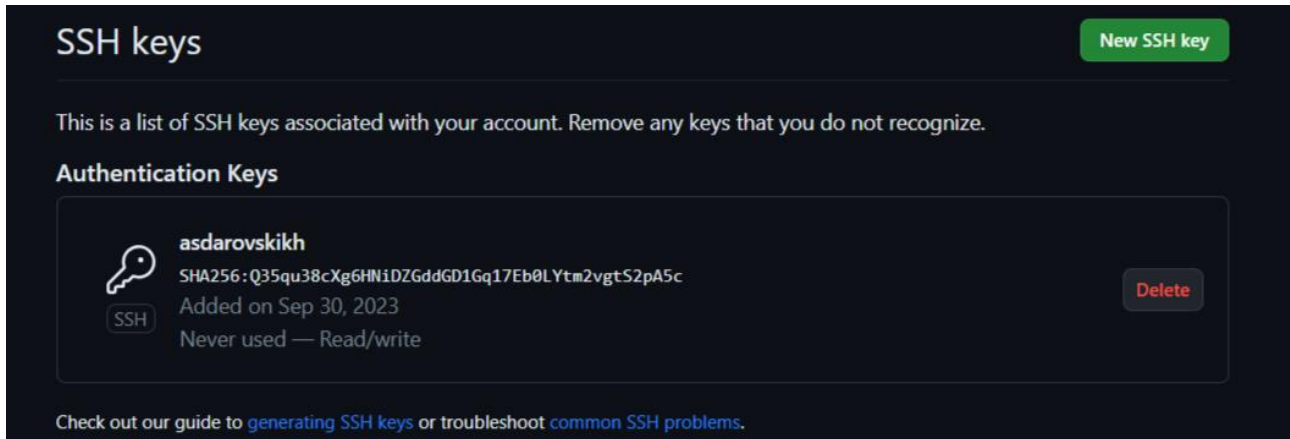


Рис. 4.7. Лист авторизованных ключей

4.4. Создание рабочего пространства и репозитория курса на основе шаблона

Переходим в терминал и создаем рабочего пространства для предмета «Архитектура Компьютера» с помощью команды `mkdir` и ключу `-p`. После чего проверяем с помощью команды `ls` (рис. 4.8)

```
[asdarovskikh@fedora ~]$ mkdir -p work/study/2023-2024/"Архитектура компьютера"
[asdarovskikh@fedora ~]$ ls ~/work/study/2023-2024
'Архитектура компьютера'
```

Рис. 4.8. Создание рабочего пространства

4.5 Создание репозитория курса на основе шаблона

Переходим в браузер на страницу репозитория с шаблоном курса по адресу <https://github.com/yamadharma/course-directory-student-template>. Далее выбираем «Use this template», чтобы использовать этот шаблон для своего репозитория (рис. 4.9)

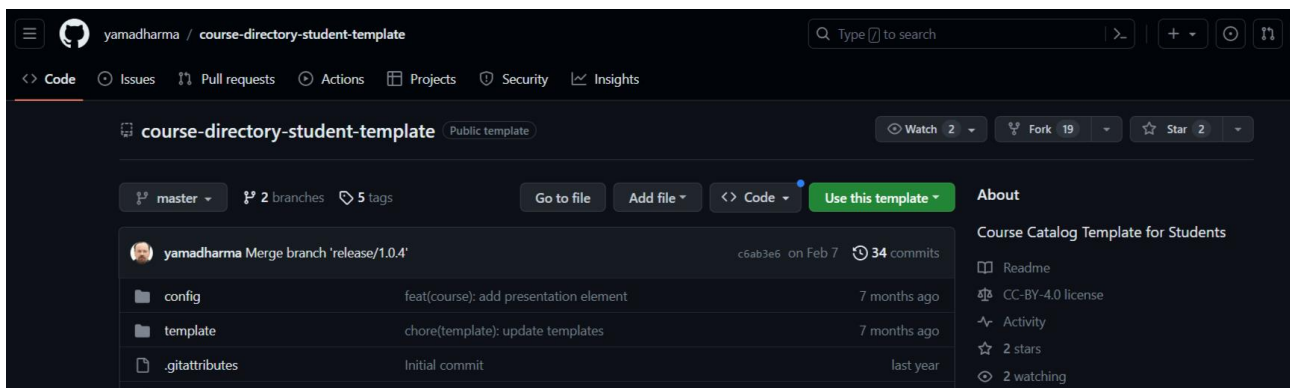


Рис. 4.9. Страница шаблона для репозитория

В открывшемся окне задаем имя репозитория `study_2022–2023_arhpc` и создаем репозиторий, нажимаю на кнопку «Create repository» (рис. 4.10.)

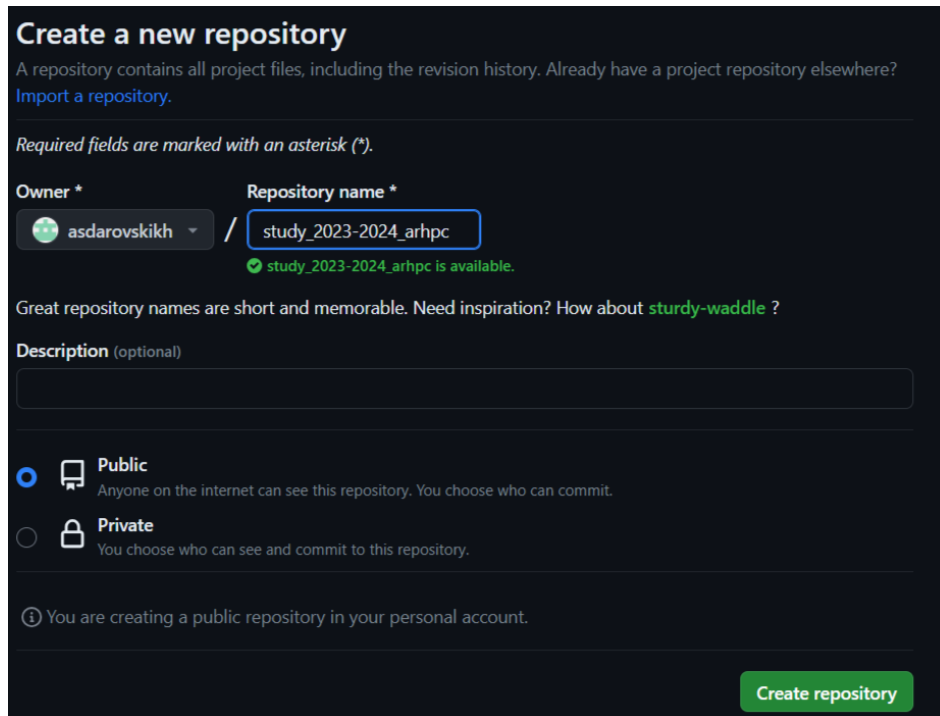


Рис. 4.10 Окно создания репозитория

Возвращаемся в терминал и переходим в созданный каталог курса с помощью команды `cd`. Клонировем созданный репозиторий с помощью команды `git clone --recursive git@github.com:study_2023–2024_arh-pc.git arch-pc` (рис. 4.11).

```
[asdarovskikh@fedora Архитектура компьютера]$ cd ~/work/study/2023-2024/"Архитектура компьютера"
[asdarovskikh@fedora Архитектура компьютера]$ git clone --recursive git@github.com:asdarovskikh/study_2023-2024_arhpc.git arch-pc
Клонирование в «arch-pc»...
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMsvHdkr4UvCOqU.
```

Рис. 4.11.

Копируем ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. 4.12)

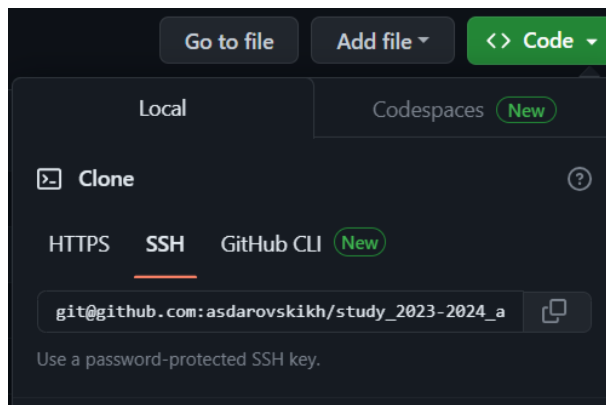


Рис. 4.12.

4.6 Настройка каталога курса

Переходим в каталог arch-pc с помощью команды `cd` и удаляем лишние файлы с помощью команды `rm` (рис. 4.13).

```
[asdarovskikh@fedora Архитектура компьютера]$ cd ~/work/study/2023-2024/Архитектура компьютера/arch-pc
[asdarovskikh@fedora arch-pc]$ rm package.json
```

Рис. 4.13. Удаление лишних файлов

Создаем необходимые каталоги (рис.4.14), отправляем созданные каталоги с локального репозитория на сервер: добавляем все созданные каталоги с помощью `git add`, комментируем и сохраняем изменения на сервере как добавление курса с помощью `git commit` (рис. 4.15).

```
[asdarovskikh@fedora arch-pc]$ echo arch-pc > COURSE
[asdarovskikh@fedora arch-pc]$ make
```

Рис. 4.14. Создание необходимых каталогов

```
[asdarovskikh@fedora arch-pc]$ git add .
[asdarovskikh@fedora arch-pc]$ git commit -am 'feat(main): make course structure'
[master bca75df] 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
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
```

Рис. 4.15. Добавление и сохранение изменений на сервере

Отправляем все на сервер с помощью push (рис. 4.15).

```
[asdarovskikh@fedora arch-pc]$ git push
Перечисление объектов: 37, готово.
Подсчет объектов: 100% (37/37), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (29/29), готово.
Запись объектов: 100% (35/35), 342.14 КиБ | 2.44 МиБ/с, готово.
Всего 35 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:asdarovskikh/study_2023-2024_arhpc.git
9207a32..bca75df master -> master
```

Рис. 4.15. Отправление файлов на сервер

Проверяем выполнение работы на самом сайте GitHub (рис. 4.16).

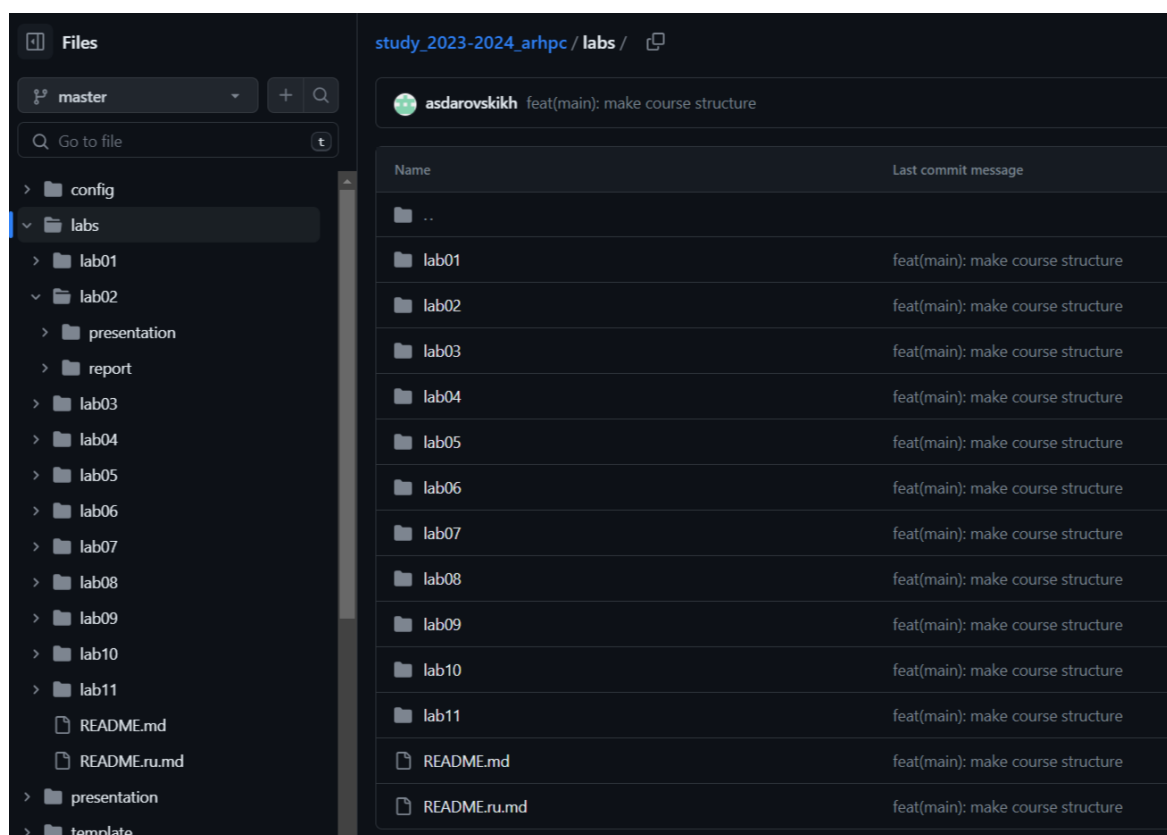


Рис. 4.16. Страница репозитория

4.7 Выполнение заданий для самостоятельной работы

1. Переходим в директорию labs/lab02/report с помощью команды cd. Создаем в каталоге файл для отчета по второй лабораторной работе с помощью команды touch (рис. 4.17).

```
[asdarovskikh@fedora arch-pc]$ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab02/report
[asdarovskikh@fedora report]$ touch Л02_Даровских_отчет.doc
```

Рис. 4.17. Создание файла

Оформляем отчет в текстовом процессоре LibreOffice Writer (рис.4.18)

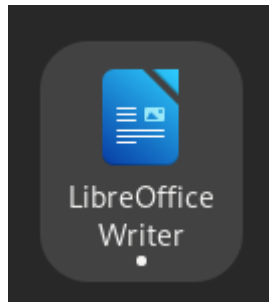


Рис. 4.18. Приложение LibreOffice Writer

После открытия текстового процессора открываю в нем созданный файл и могу начать в нем работу над отчетом (рис. 4.19).

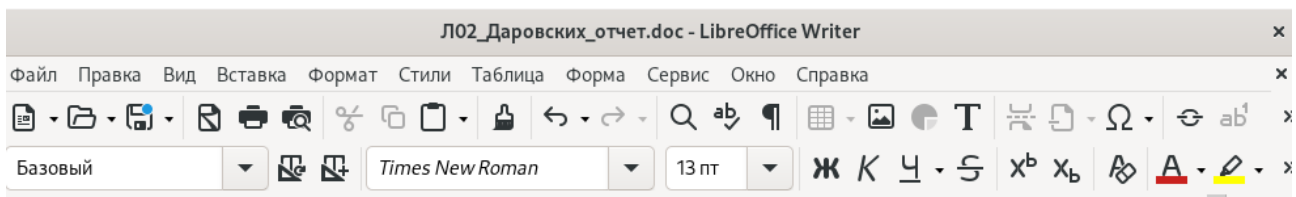


Рис. 4.19. Работа с отчетом в текстовом процессоре

2. Первая лабораторная работа у меня находится в подкаталоге домашней директории Загрузки. Перехожу в каталог `/arch-pc/labs/lab01/report` и копирую сюда файл `Л01_Даровских_отчет` с помощью команды `cp` и проверяю с помощью команды `ls`. (рис. 4.20)

```
[asdarovskikh@fedora ~]$ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab02/report
[asdarovskikh@fedora report]$ cp ~/Загрузки/Л01_Даровских_отчет.pdf /home/asdarovskikh/work/study/2023-2024/"Архитектура
компьютера"/arch-pc/labs/lab01/report
[asdarovskikh@fedora report]$ ls
bib image Makefile pandoc report.md Л01_Даровских_отчет.pdf
```

Рис. 4.20. Копирование отчета по лабораторной работе в каталог

3. Переходим в каталог `arch-pc`, чтобы отправить все добавленные файлы в репозиторий. Используем команды `git add` (добавление всех измененных файлов) и `git commit -m «Add existing files (сохранение всех измененных файлов)»`, а также `git push`, чтобы отправить все изменения на сервер (рис. 4.21.)

```
[asdarovskikh@fedora report]$ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc
[asdarovskikh@fedora arch-pc]$ git add .
[asdarovskikh@fedora arch-pc]$ git commit -m "Add existing files"
[master 391a08a] Add existing files
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 labs/lab01/report/Л01_Даровских_отчет.pdf
create mode 100644 labs/lab02/report/Л02_Даровских_отчет.doc
[asdarovskikh@fedora arch-pc]$ git push
Перечисление объектов: 14, готово.
Подсчет объектов: 100% (12/12), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (8/8), готово.
Запись объектов: 100% (8/8), 826.29 Киб | 5.37 Миб/с, готово.
Всего 8 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:asdarovskikh/study_2023-2024_arhpc.git
 bca75df..391a08a master -> master
```

Рис. 4.21 Добавление и отправка файлов в центральный репозиторий

Проверим выполнение работы на сайте github (рис.4.22), (рис.4.23).

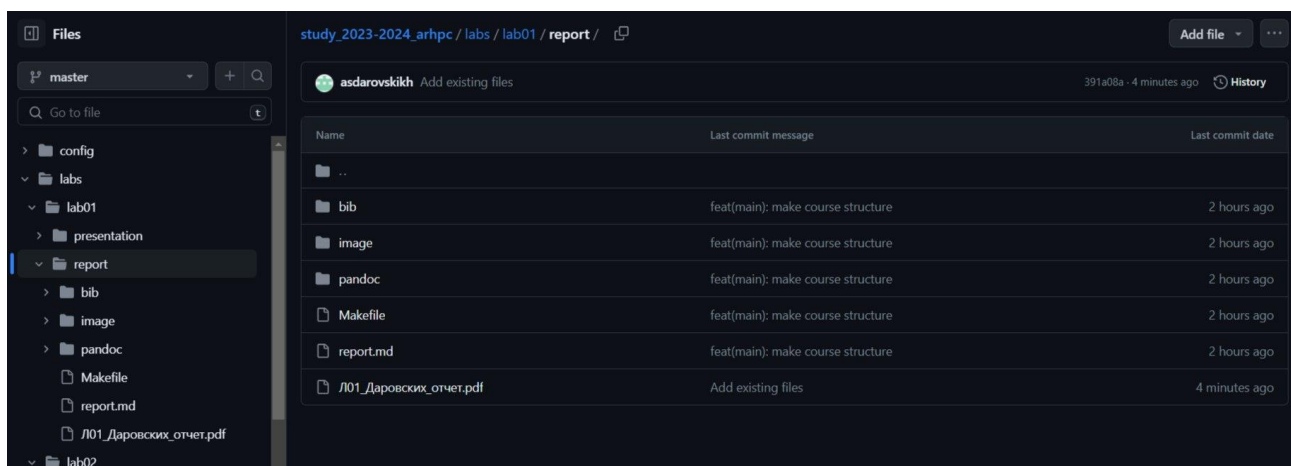


Рис. 4.22.Страница с добавленным файлом в каталоге lab01

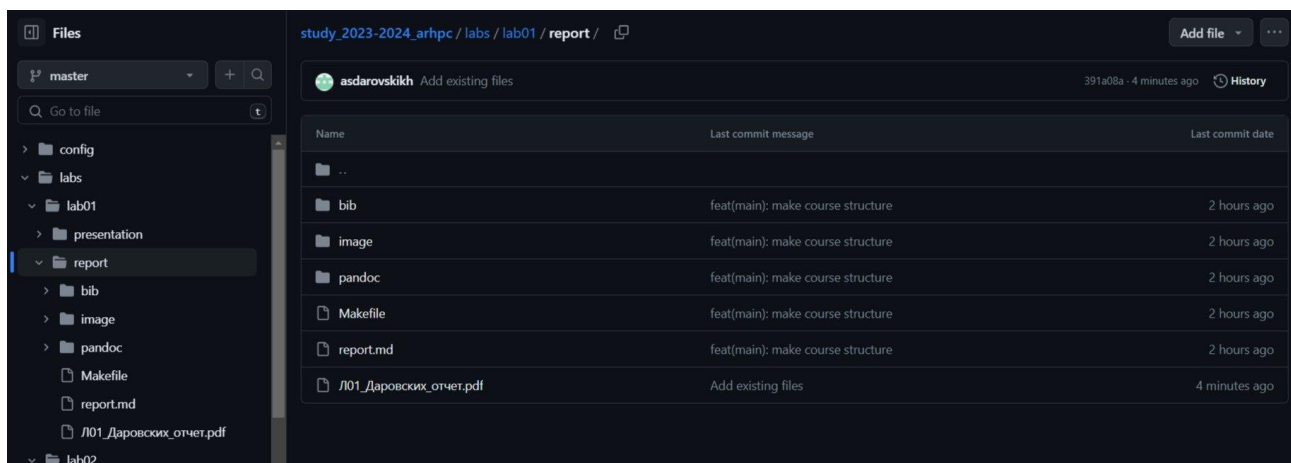


Рис. 4.23.Страница с добавленным файлом в каталоге lab02

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.

Список литературы

1. https://esystem.rudn.ru/pluginfile.php/2089082/mod_resource/content/0/Лабораторная%20работа%20№2.%20Система%20контроля%20версий%20Git.pdf

