

Отчёт по лабораторной работе №2

Дисциплина: архитектура компьютера.

Наговицын Арсений Владимирович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Настройка GitHub:	8
3.2	Базовая настройка Git:	9
3.3	Создание SSH-ключа:	11
3.4	Создание рабочего пространства и репозитория курса на основе шаблона:	12
3.5	Создание репозитория курса на основе шаблона:	13
3.6	Настройка каталога курса:	15
3.7	Выполнение заданий для самостоятельной работы:	17
4	Выводы	22

Список иллюстраций

3.1	Аккаунт GitHub.	9
3.2	Предварительная конфигурация git.	10
3.3	Настройка кодировки.	10
3.4	Настройка кодировки.	10
3.5	Параметр autocrlf.	10
3.6	Параметр safecrlf.	10
3.7	Генерация SSH-ключа.	11
3.8	Копирование содержимого файла.	11
3.9	Окно SSH and GPG keys.	12
3.10	Добавление ключа.	12
3.11	Создание рабочего пространства.	12
3.12	Страница шаблона для репозитория.	13
3.13	Окно создания репозитория.	13
3.14	Созданный репозиторий.	14
3.15	Перемещение между директориями.	14
3.16	Окно с ссылкой для копирования репозитория.	15
3.17	Клонирование репозитория.	15
3.18	Перемещение между директориями.	16
3.19	Удаление файлов.	16
3.20	Создание каталогов.	16
3.21	Добавление и сохранение изменений на сервере.	16
3.22	Выгрузка изменений на сервер.	17
3.23	Страница репозитория.	17
3.24	Создание файла.	17
3.25	Меню приложений.	18
3.26	Работа с отчетом в текстовом редакторе.	18
3.27	Перемещение между директориями.	19
3.28	Проверка местонахождения файла.	19
3.29	Копирование файла.	19
3.30	Добавление файла на сервер.	19
3.31	Сохранение изменений.	19
3.32	Отправка в центральный репозиторий сохраненных изменений.	20
3.33	Страница каталога в репозитории.	20
3.34	Каталог lab01/report.	20
3.35	Каталог lab02/report.	21

Список таблиц

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. # Выполнение лабораторной работы

3.1 Настройка GitHub:

Так как мой аккаунт на GitHub уже создан перехожу к следующему заданию (рис. 3.1).



Рис. 3.1: Аккаунт GitHub.

3.2 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. 3.2).

```
[avnagovicihn@fedora ~]$ git config --global user.name "<Arseniy Nagovitsyn>"  
[avnagovicihn@fedora ~]$ git config --global user.email "<1132239111@pfur.ru>"
```

Рис. 3.2: Предварительная конфигурация git.

Настраиваю utf-8 в выводе сообщений git для корректного отображения символов(рис. 3.3).

```
[avnagovicihn@fedora ~]$ git config --global core.quotePath false
```

Рис. 3.3: Настройка кодировки.

Задаю имя «master» для начальной ветки(рис. 3.4).

```
[avnagovicihn@fedora ~]$ git config --global init.defaultBranch master
```

Рис. 3.4: Настройка кодировки.

Задаю параметр autocrlf со значением input, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах (рис. 5). CR и LF – это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах.(рис. 3.5)

```
[avnagovicihn@fedora ~]$ git config --global core.autocrlf input
```

Рис. 3.5: Параметр autocrlf.

Задаю параметр safecrlf со значением warn, так Git будет проверять преобразование на обратимость (рис. 6). При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации (рис. 3.6)

```
[avnagovicihn@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 3.6: Параметр safecrlf.

3.3 Создание SSH-ключа:

Для последующей идентификации пользователя на сервере репозитория 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. Ключ автоматически сохранится в каталоге `~/.ssh/` (рис. 3.7).

```
[avnagovicihn@fedora ~]$ ssh-keygen -C "Arseniy Nagovitsyn <1132239111@pfur.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/avnagovicihn/.ssh/id_rsa):
Created directory '/home/avnagovicihn/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/avnagovicihn/.ssh/id_rsa
Your public key has been saved in /home/avnagovicihn/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:MvZ2z6y6e7sjk2ecJ7CMRON4BSSdohaH6psJevB1IpE Arseniy Nagovitsyn <1132239111@pfur.ru>
The key's randomart image is:
+---[RSA 3072]-----+
|  ..O.. |
|  O O.+ |
|  . = . . |
|  . E  O . |
|  . . . ++oS |
|  O.. +.=+. |
|  oo+o = oo=. |
|  .+O  ..*.O+. |
|  .      +X+B+ |
+---[SHA256]-----+
```

Рис. 3.7: Генерация SSH-ключа.

Копирую ключ из открытой директории, в которой он был сохранен (рис. 3.8).

```
[avnagovicihn@fedora ~]$ cat ~/.ssh/id_rsa.pub
```

Рис. 3.8: Копирование содержимого файла.

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» (рис. 3.9).

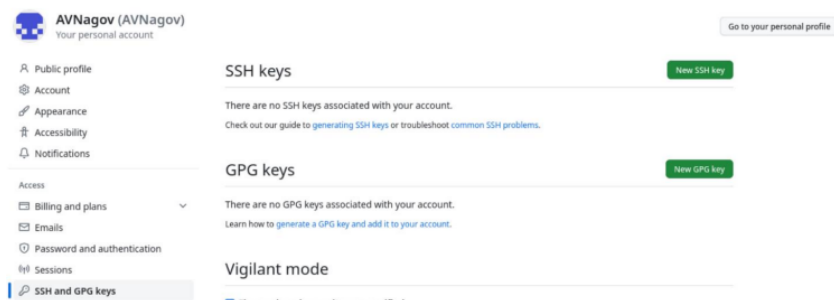


Рис. 3.9: Окно SSH and GPG keys.

Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы завершить добавление ключа. После чего получаю такое сообщение (рис. 3.10).



Рис. 3.10: Добавление ключа.

3.4 Создание рабочего пространства и репозитория курса на основе шаблона:

Открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты mkdir, благодаря ключу -p создаю все директории после домашней ~/work/study/2023-2024/“Архитектура компьютера”. Далее проверяю с помощью ls, были ли созданы необходимые мне каталоги (рис. 3.11).

```
[avnagovicihn@fedora ~]$ mkdir -p work/study/2023-2024/"Архитектура компьютера."
[avnagovicihn@fedora ~]$ ls
work  Документы  Изображения  Общедоступные  Шаблоны
Видео  Загрузки  Музыка  'Рабочий стол'
```

Рис. 3.11: Создание рабочего пространства.

3.5 Создание репозитория курса на основе шаблона:

В браузере перехожу на страницу репозитория с шаблоном курса. Далее выбираю «Use this template», чтобы использовать этот шаблон для своего репозитория (рис. 3.12).

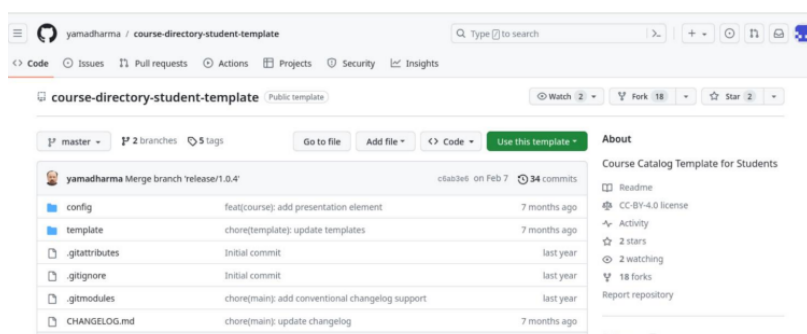


Рис. 3.12: Страница шаблона для репозитория.

В открывшемся окне задаю имя репозитория (Repository name): study_2023–2024_arh-рс и создаю репозиторий, нажав на кнопку «Create repository from template» (рис. 3.13).

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * AVNagov / Repository name *

study_2023-2024_arh-pc is available.

Great repository names are short and memorable. Need inspiration? How about [refactored-broccoli](#) ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

① You are creating a public repository in your personal account.

Рис. 3.13: Окно создания репозитория.

Репозиторий создан(рис. 3.14).

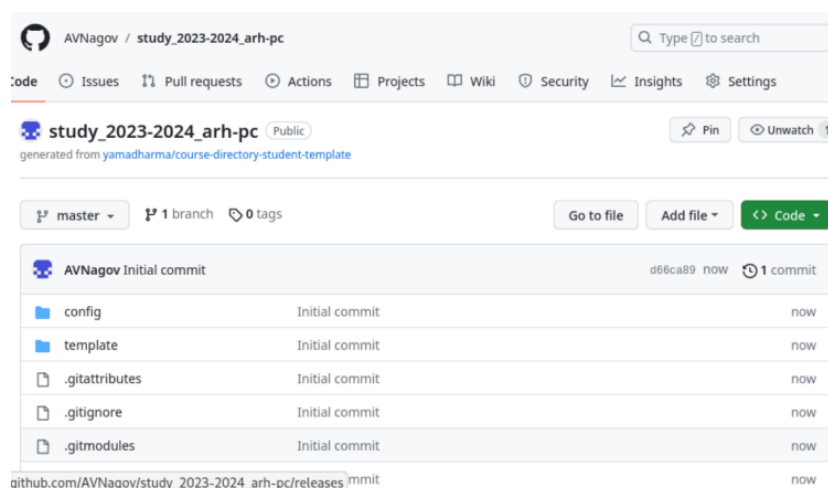


Рис. 3.14: Созданный репозиторий.

Через терминал перехожу в созданный каталог курса (рис. 3.15).

```
[avnagovic@fedora ~]$ cd ~/work/study/2023-2024/'Архитектура компьютера.'
```

Рис. 3.15: Премещение между директориями.

Копирую ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. 3.16).

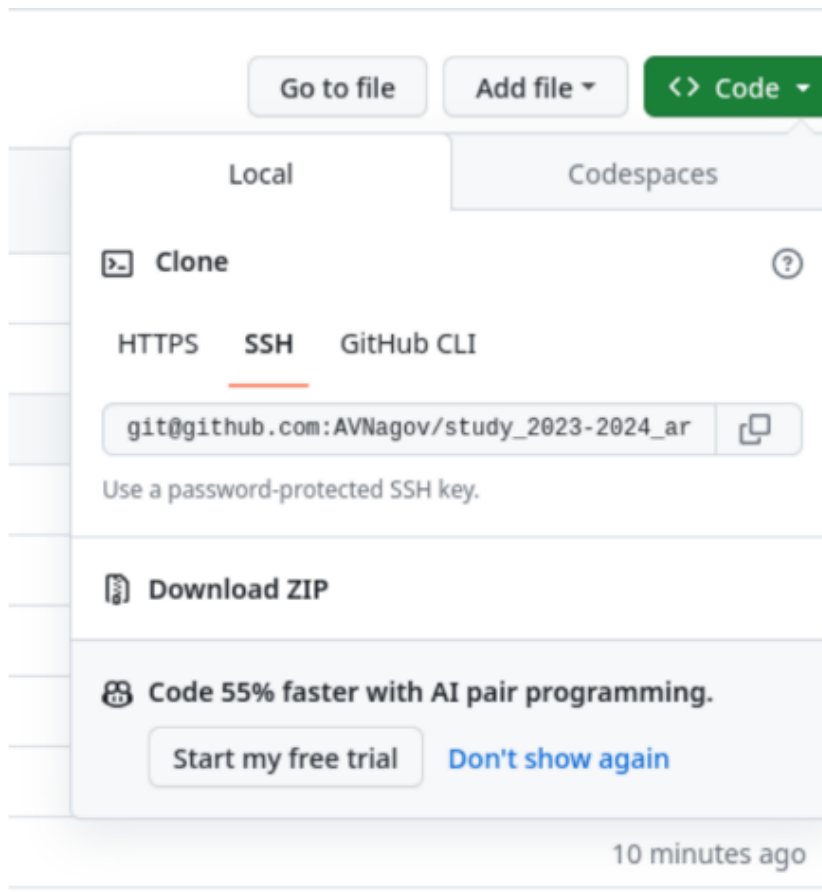


Рис. 3.16: Окно с ссылкой для копирования репозитория.

Клонирую созданный репозиторий с помощью команды `git clone --recursive git@github.com:AVNagov/study_2022–2023_arh-pc.git arch-pc` (рис. 3.17).

```
[avnagov@localhost ~]$ git clone --recursive git@github.com:AVNagov/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), 16.93 Киб | 16.93 Мб/с, готово.
```

Рис. 3.17: Клонирование репозитория.

3.6 Настройка каталога курса:

Перехожу в каталог `arch-pc` (рис. 3.18).

```
[avnagovicihn@fedora Архитектура компьютера]$ cd ~/work/study/2023-2024/'Архитектура компьютера.'/arch-pc
```

Рис. 3.18: Перемещение между директориями.

Удаляю лишние файлы с помощью утилиты `rm`(рис. 3.19).

```
[avnagovicihn@fedora arch-pc]$ rm package.json
```

Рис. 3.19: Удаление файлов.

Создаю необходимые каталоги (рис. 3.20).

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

Рис. 3.20: 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. 3.21).

```
[avnagovicihn@fedora arch-pc]$ git add .
[avnagovicihn@fedora arch-pc]$ git commit -am 'feat(main): make ciourse structure'
[master 6ce38c9] feat(main): make ciourse 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
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_tablenos.py
```

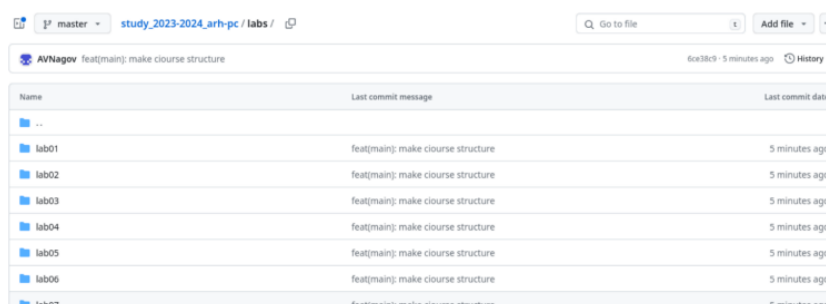
Рис. 3.21: Добавление и сохранение изменений на сервере.

Отправляю все на сервер с помощью `push` (рис. 3.22).


```
[avnagovicihn@fedora arch-pc]$ git push
Перечисление объектов: 37, готово.
Подсчет объектов: 100% (37/37), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (29/29), готово.
Запись объектов: 100% (35/35), 342.14 КиБ | 2.53 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:AVNagov/study_2023-2024_arh-pc.git
d66ca89..6ce38c9 master -> master
```

Рис. 3.22: Выгрузка изменений на сервер.

Проверяю правильность выполнения работы сначала на самом сайте GitHub (рис. 3.23).



Name	Last commit message	Last commit date
..		
lab01	feat(main): make course structure	5 minutes ago
lab02	feat(main): make course structure	5 minutes ago
lab03	feat(main): make course structure	5 minutes ago
lab04	feat(main): make course structure	5 minutes ago
lab05	feat(main): make course structure	5 minutes ago
lab06	feat(main): make course structure	5 minutes ago
lab07	feat(main): make course structure	5 minutes ago

Рис. 3.23: Страница репозитория.

3.7 Выполнение заданий для самостоятельной работы:

1. Перехожу в директорию labs/lab02/report с помощью утилиты cd. Создаю в каталоге файл для отчета по второй лабораторной работе (рис. 3.24).

```
[avnagovicihn@fedora arch-pc]$ cd ~/work/study/2023-2024/'Архитектура компьютера.'/arch-pc/labs/lab02/report
[avnagovicihn@fedora report]$ touch Л02_Наговицын_Отчет
```

Рис. 3.24: Создание файла.

Оформить отчет я смогу в текстовом процессоре LibreOffice Writer, найдя его в открытом с помощью (рис. 3.25).

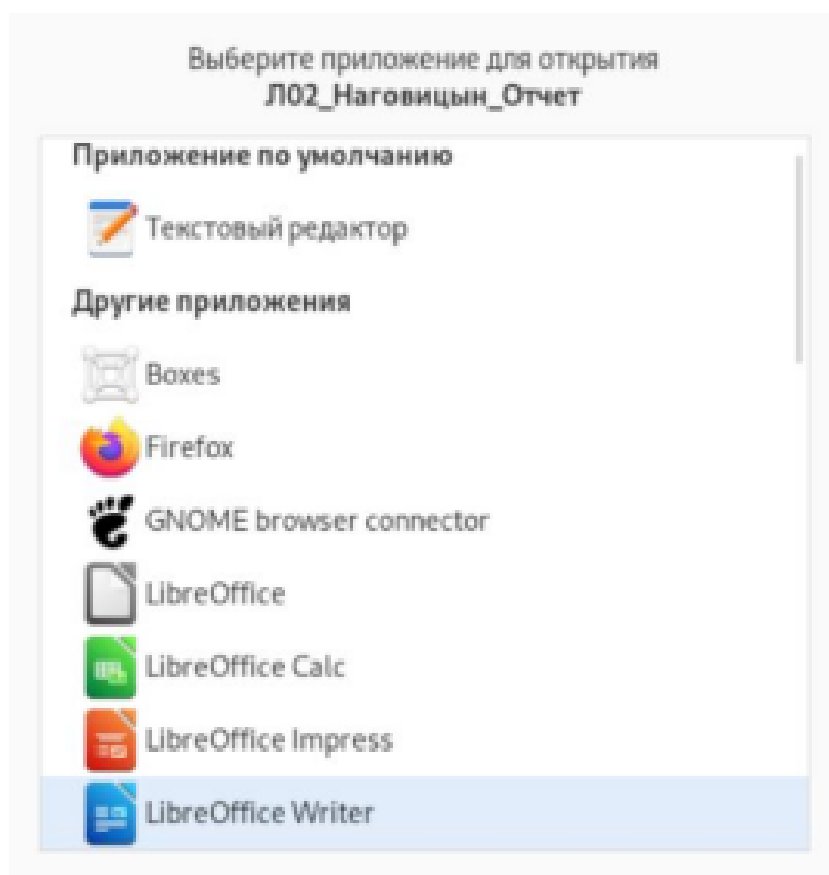


Рис. 3.25: Меню приложений.

После открытия текстового процессора открываю в нем созданный файл и могу начать в нем работу над отчетом (рис. 3.26).

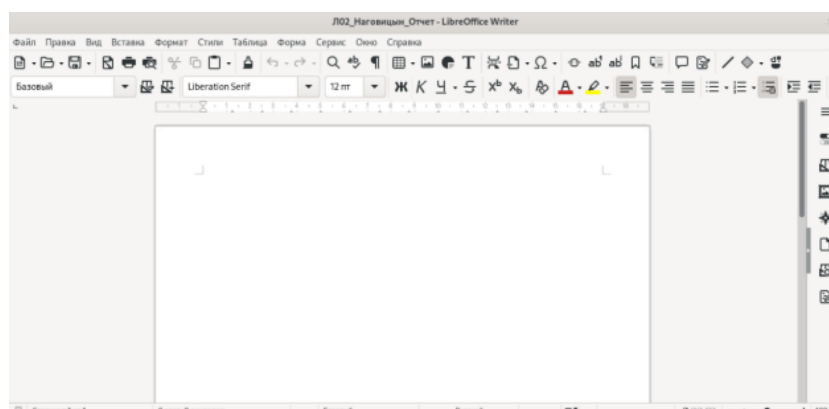


Рис. 3.26: Работа с отчетом в текстовом редакторе.

2. Перехожу из подкаталога lab03/report в подкаталог lab01/report (рис. 3.27).

```
[avnagovicihn@fedora report]$ cd ..  
[avnagovicihn@fedora lab02]$ cd ..  
[avnagovicihn@fedora labs]$ cd lab01/report
```

Рис. 3.27: Перемещение между директориями.

Проверяю местонахождение отчета по первой лабораторной работе (рис. 3.28).

```
[avnagovicihn@fedora report]$ ls ~/Загрузки  
Л01_Наговицын_Отчет.pdf
```

Рис. 3.28: Проверка местонахождения файла.

Копирую отчет по лабораторной работе с помощью утилиты cp(рис. 3.29).

```
[avnagovicihn@fedora lab01]$ cp ~/Загрузки/Л01_Наговицын_Отчет.pdf /home/avnagovicihn/work/study/2023-2024/Архитектура_компьютера./arch-pc/labs/lab01/report
```

Рис. 3.29: Копирование файла.

3. Добавляю с помощью команды git add созданный файл: Л01_Наговицын_Отчет (рис. 3.30).

```
[avnagovicihn@fedora report]$ git add Л01_Наговицын_Отчет.pdf
```

Рис. 3.30: Добавление файла на сервер.

Сохраняю все добавленные изменения(рис. 3.31).

```
[avnagovicihn@fedora report]$ git commit -m "Add existing file"  
[master a6f6fa0] Add existing file  
1 file changed, 0 insertions(+), 0 deletions(-)
```

Рис. 3.31: Сохранение изменений.

Отправляю в центральный репозиторий изменения командой `git push -f origin master` (рис. 3.32).

```
[avnagovic@fedora report]$ git push -f origin master
Перечисление объектов: 10, готово.
Подсчет объектов: 100% (10/10), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.28 МиБ | 9.56 МиБ/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:AVNagov/study_2023-2024_arh-pc.git
6ce38c9..a6f6fa0 master -> master
```

Рис. 3.32: Отправка в центральный репозиторий сохраненных изменений.

Проверяю на сайте GitHub правильность выполнения заданий. Вижу, что пояснение к совершенным действиям отображается (рис. 3.33).

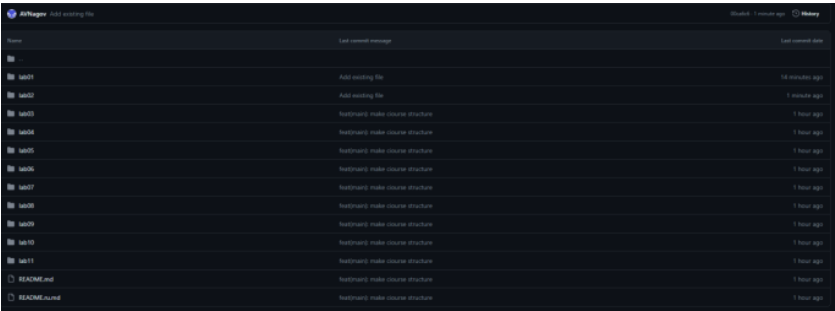


Рис. 3.33: Страница каталога в репозитории.

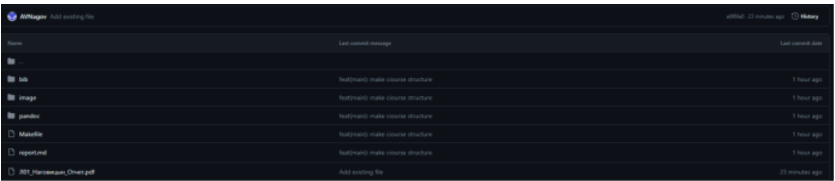


Рис. 3.34: Каталог lab01/report.

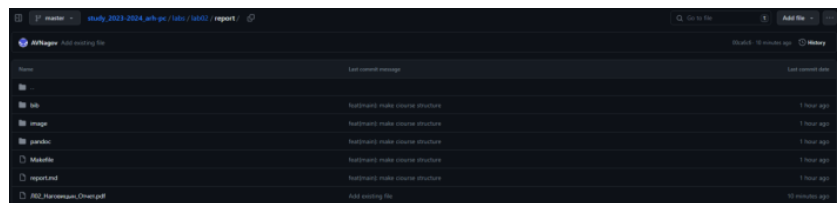


Рис. 3.35: Каталог lab02/report.

4 Выводы

При выполнении данной лабораторной работы я изучил идеологию и применение средств контроля версий, а также приобрел практические навыки по работе с системой git.