

# 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: Тарасова Алина

Группа: НКАбд 05-23

МОСКВА

## **Содержание**

1. Цель работы.....	3
2. Задание.....	4
3. Теоретическое введение.....	5
4. Выполнение лабораторной работы.....	7
5. Выводы.....	17
6. Список литературы.....	18

## **1 Цель работы**

Целью работы является изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

## 2 Задания

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

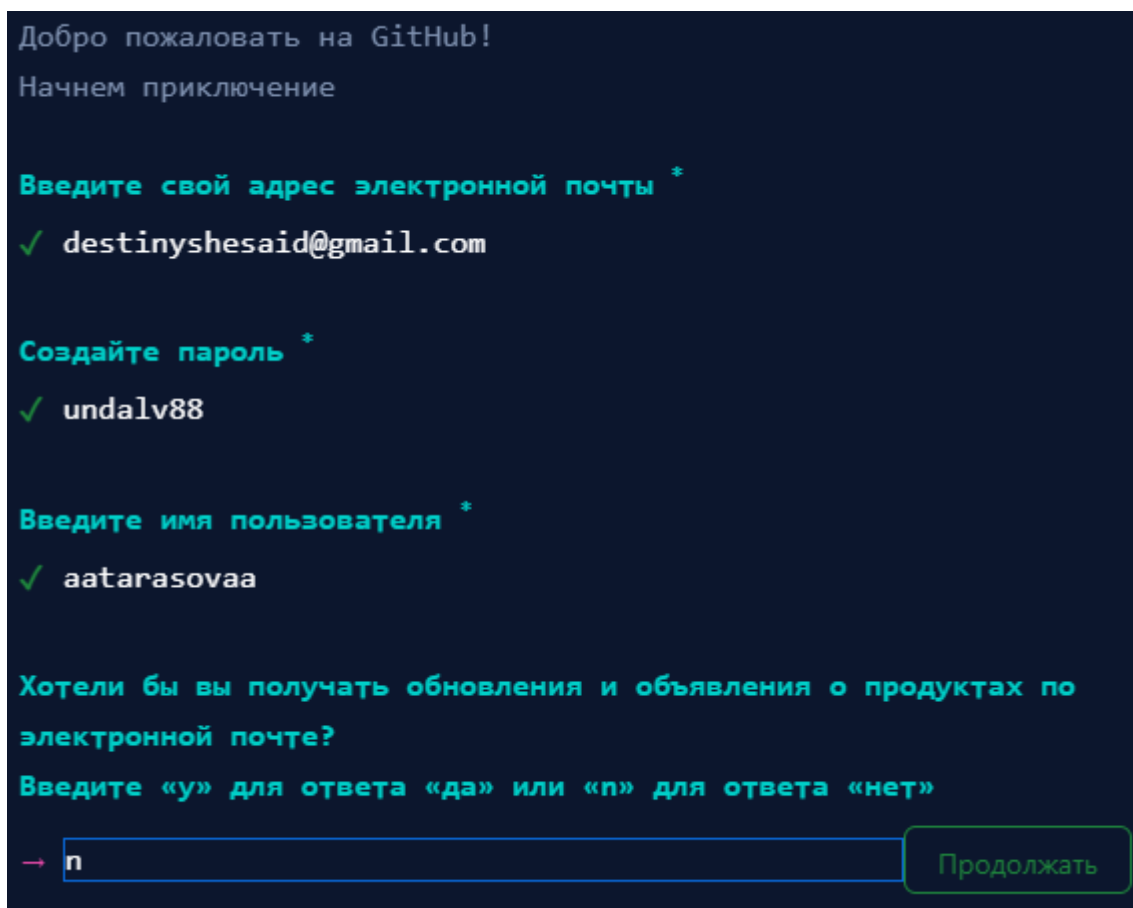
Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля 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

Создаю учетную запись на сайте GitHub (рис. 4.1). Далее я заполнила основные данные учетной записи.



Добро пожаловать на GitHub!  
Начнем приключение

Введите свой адрес электронной почты \*

✓ destinyshesaid@gmail.com

Создайте пароль \*

✓ undalv88

Введите имя пользователя \*

✓ aatarasovaa

Хотели бы вы получать обновления и объявления о продуктах по электронной почте?

Введите «у» для ответа «да» или «н» для ответа «нет»

→

Рис. 4.1: Заполнение данных учетной записи GitHub

Аккаунт создан (рис. 4.2).

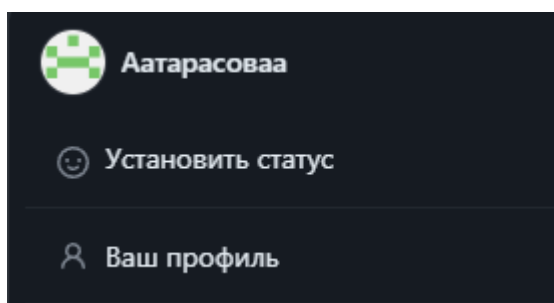


Рис. 4.2: Аккаунт GitHub

## 4.2. Базовая настройка Git

Сначала сделаю предварительную конфигурацию git. Открываю терминал и 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).

```
aatarasova@aatarasova-VirtualBox:~$ git config --global user.name "<Alina Tarasova>"
aatarasova@aatarasova-VirtualBox:~$ git config --global user.email "<1132236013@rudn.ru>"
```

Рис. 4.3: Предварительная конфигурация git

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

```
aatarasova@aatarasova-VirtualBox:~$ git config --global core.quotePath false
```

Рис. 4.4: Настройка кодировки

Задаю имя «master» для начальной ветки, параметры `autocrlf` и `safecrlf` (рис. 4.5):

```
aatarasova@aatarasova-VirtualBox:~$ git config --global init.defaultBranch master
aatarasova@aatarasova-VirtualBox:~$ git config --global core.autocrlf input
aatarasova@aatarasova-VirtualBox:~$ git config --global core.safecrlf warn
```

Рис. 4.5: Создание имени для начальной ветки, параметров `autocrlf` и `safecrlf`.

## 4.3. Создание SSH ключа

Для последующей идентификации на сервере репозитория генерирую пару ключей (приватный и открытый). Ключи сохраняются в каталоге `~/.ssh/` (рис.4.6)

```
aatarasova@aatarasova-VirtualBox:~$ ssh-keygen -C"Alina Tarasova<1132236013@rudn.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aatarasova/.ssh/id_rsa):
/home/aatarasova/.ssh/id_rsa already exists.
```

Рис.4.6 Ключ в каталоге `~/.ssh/`.



Далее необходимо загрузить сгенерированный открытый ключ. Для этого захожу на сайт <http://github.org/> под своей учётной записью и перехожу в меню Setting. После этого выбираю в боковом меню SSH and GPG keys и нажимаю кнопку New SSH key.

Xclip – утилита, позволяющая скопировать любой текст через терминал. Оказывается, в дистрибутиве Linux ее сначала надо установить. Устанавливаю xclip с помощью команды apt-get install с ключом -y от имени суперпользователя, введя в начале команды sudo (рис.4.7.).

```
aatarasova@aatarasova-VirtualBox:~$ sudo apt-get install -y xclip
```

Рис.4.7. Установка xclip

Скопировав из локальной консоли ключ в буфер обмена cat ~/.ssh/id\_rsa.pub | xclip -sel clip (рис.4.8), вставляю ключ в появившееся на сайте поле и указываем для ключа имя aatarasova2309 (рис.4.9).

```
aatarasova@aatarasova-VirtualBox:~$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

Рис. 4.8. Копирование ключа в буфер обмена.

Добавить новый ключ SSH

Заголовок

aatarasova2309

Тип ключа

Ключ аутентификации

Ключ

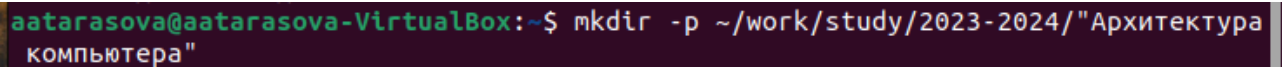
```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGC05Z0tZTY0P+mFDZ1ZfYl6xRm9aHTI3jc62OKYfK+YE8itJYZqloZ4czkGSfQSm
gslP2rSyslPsfjRZW0zQfr2ffqm6xXkQ3U7o+JEM9PG3rWXbfEwjTmHrJ+25X1WxboDILa5zl+PK+8HoVipWHKBn0wB+yGsE
w2cTkoXrguA6v1dbfCQK9n4WMValu1keu8oo/kXOZRqjAyahTrQR55qe3VgaxUjn3o53OofeR/G4w9nW4VqQhwtl8jSTIXTJ/t
uoittYg7XZEz+vL81xUL2Lg36HRt3ZSxVg/lqCVL7WNoSayxSZs9hMVDNWEq/3LWcO1ywmKv2zJgfBtsGMJuUF8DjJN/XT/bL
zev0V1oEUz8mSM5PKoeL/kekwxwff/b2xsCbbwTEuSUWoMg5xpmsxLKbrCJJKi+2d1cPNFwpf51LSUlySeoGkzQbsXZa54sV
daZk0HPuHLgelhTASefKdnJhX//i2Z8xBSEqQMKvvXOXw5hvM8kNqSeh3LWU= Alina Tarasova <1132236013@rudn.ru>
```

Добавить SSH-ключ

Рис.4.9. Создание ключа SSH.

#### **2.4.4. Создание рабочего пространства и репозитория курса на основе шаблона**

Открываю терминал и создаю каталог для предмета «Архитектура компьютера» (рис.4.10):



```
aatarasova@aatarasova-VirtualBox:~$ mkdir -p ~/work/study/2023-2024/"Архитектура компьютера"
```

Рис.4.10. Создание каталога для предмета «Архитектура компьютера» с помощью команды mkdir и ключа -p.

#### **2.4.5. Сознание репозитория курса на основе шаблона**

Репозиторий на основе шаблона можно создать через web-интерфейс github. Перехожу на страницу репозитория с шаблоном курса <https://github.com/yamadharmacourse-directory-student-template>. Далее выбираю Use this template (рис.4.11).

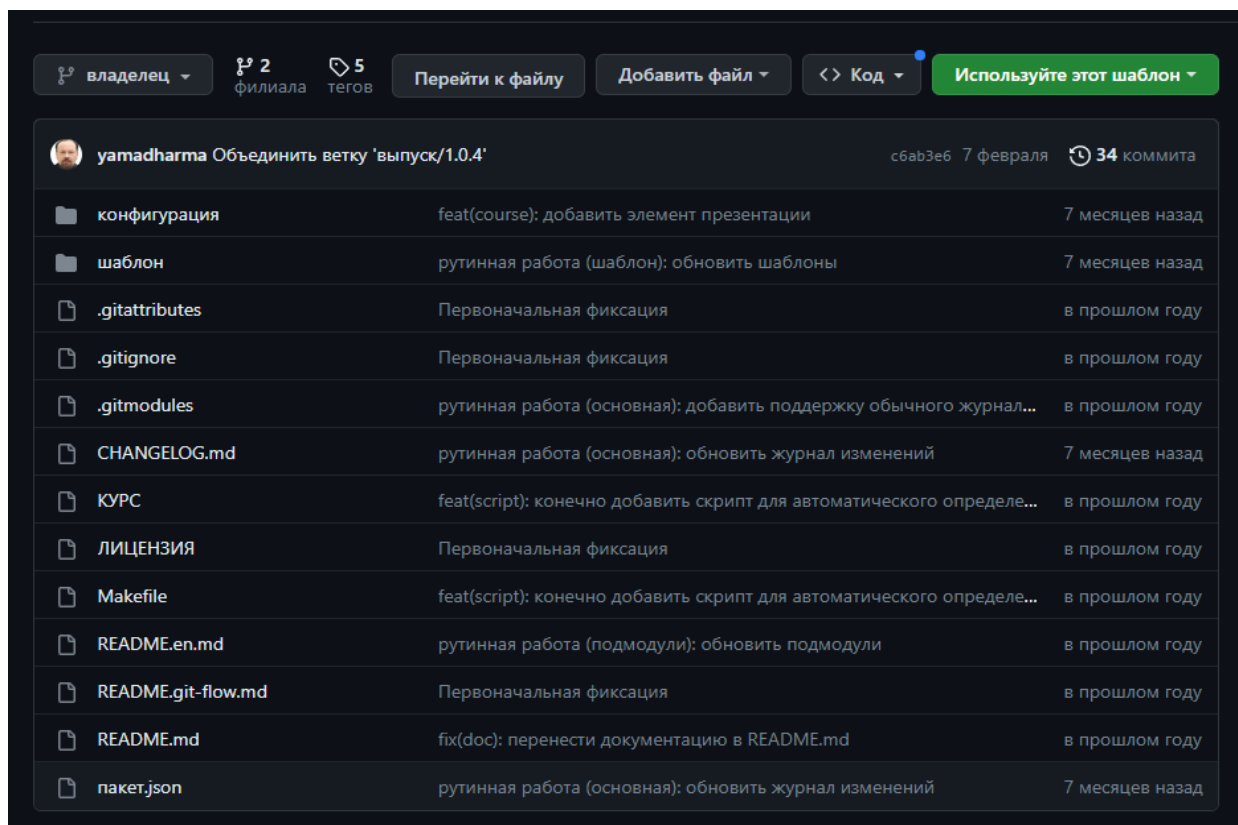


Рис. 4.11. Выбор шаблона


В открывшемся окне задаю имя репозитория study\_2023–2024\_arhpc и создаю репозиторий (рис.4.12).

## Создать новый репозиторий

Репозиторий содержит все файлы проекта, включая историю изменений. У вас уже есть репозиторий проекта в другом месте? [Импортируйте репозиторий.](#)

Обязательные поля отмечены звездочкой (\*).


Владелец \* / Имя репозитория \*


 Аатарасова / study\_2023–2024\_arhpc


⚠ Ваш новый репозиторий будет создан как Study\_2023-2024\_arhpc .  
Имя репозитория может содержать только буквы ASCII, цифры и символы ., и - \_

Имена хороших репозитория короткие и запоминающиеся. Вам нужно вдохновение? Как насчет **пушистая окто-ложка ?**

Описание (необязательно)

☒  **Общественный**  
Любой человек в Интернете может увидеть этот репозиторий. Вы выбираете, кто может совершить.

☐  **Частный**  
Вы выбираете, кто может видеть и сохранять этот репозиторий.

 Вы создаете публичный репозиторий в личном кабинете.

[Создать репозиторий](#)

Рис.4.12. Создание репозитория study\_2023–2024\_arhpc.

Открываю терминал и перехожу в каталог курса (рис.4.13):

```
aatarasova@aatarasova-VirtualBox:~$ cd ~/work/study/2023-2024/"Архитектура компьютера"
aatarasova@aatarasova-VirtualBox:~/work/study/2023-2024/Архитектура компьютера$
```

Рис. 4.13. Перемещение между директориями.

Клонирую созданный репозиторий с помощью команды git clone – recursive:

```
aatarasova@aatarasova-VirtualBox:~/work/study/2023-2024/Архитектура компьютера$ git clone --recursive git@github.com:aatarasovaa/study_2023-2024_arhpc.git arch-
pc
Клонирование в «arch-pc»...
```

Рис. 4.14. Клонирование созданного репозитория.

Копирую ссылку для клонирования на странице созданного

репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. 4.15).

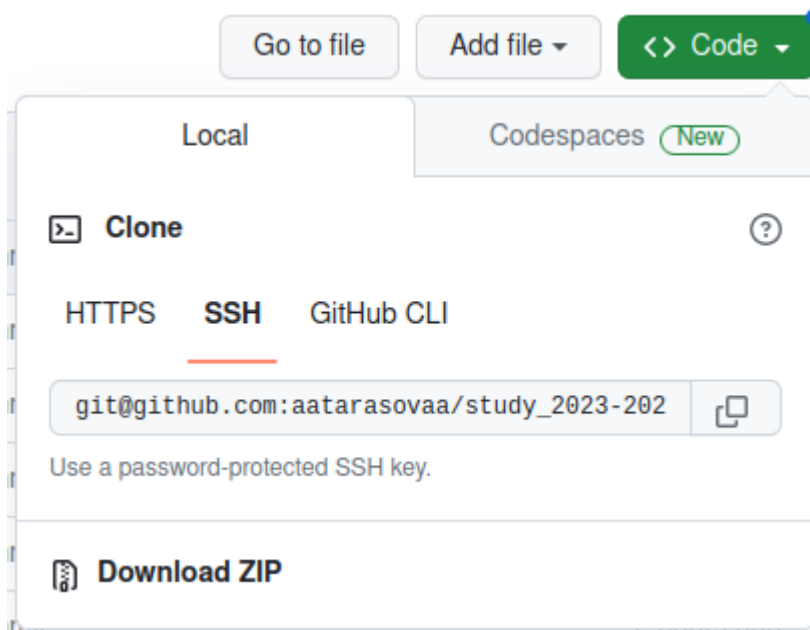


Рис. 4.15. Окно с ссылкой для копирования репозитория.

## 2.4.6. Настройка каталога курса

Перехожу в каталог курса (рис.4.16):

```
aatarasova@aatarasova-VirtualBox:~/work/study/2023-2024/Архитектура компьютера$  
cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc
```

Рис.4.16. Перемещение между директориями.

Удаляю лишние файлы (рис.4.17):

```
aatarasova@aatarasova-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/a  
rch-pc$ rm package.json
```

Рис.4.17. Удаление лишних файлов с помощью утилиты rm.

Создаю необходимые каталоги:

```
aatarasova@aatarasova-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/a  
rch-pc$ echo arch-pc > COURSE  
aatarasova@aatarasova-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/a  
rch-pc$ make
```

Рис.4.18. Создание каталогов.

Отправляю файлы на сервер: добавляю созданные файлы с помощью команды `git add` (рис.4.19).

```
aatarasova@aatarasova-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/a  
rch-pc$ git add .
```

Рис.4.19. Добавление изменений на сервере.

Сохраняю все добавленные изменения и все изменённые файлы с помощью команды `git commit -am` (рис.4.20).

```
rch-pc$ git commit -am 'feat(main): make course structure'  
[master fd4c55f] 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  
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  
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/main.py  
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/pandocattributes.py  
create mode 100644 labs/lab01/report/report.md  
create mode 100644 labs/lab02/presentation/Makefile  
create mode 100644 labs/lab02/presentation/image/kulyabov.jpg  
create mode 100644 labs/lab02/presentation/presentation.md  
create mode 100644 labs/lab02/report/Makefile  
create mode 100644 labs/lab02/report/bib/cite.bib  
create mode 100644 labs/lab02/report/image/placeimg_800_600_tech.jpg  
create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl  
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_eqnos.py  
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_fignos.py  
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_secnos.py  
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_tablenos.py  
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/__init__.py  
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/core.py  
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/main.py  
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/pandocattributes.py  
create mode 100644 labs/lab02/report/report.md
```

Рис. 4.20. Сохранение изменений на сервере.

Отправляю все произведённые изменения на сервер с помощью команды `git push` (рис.4.21).

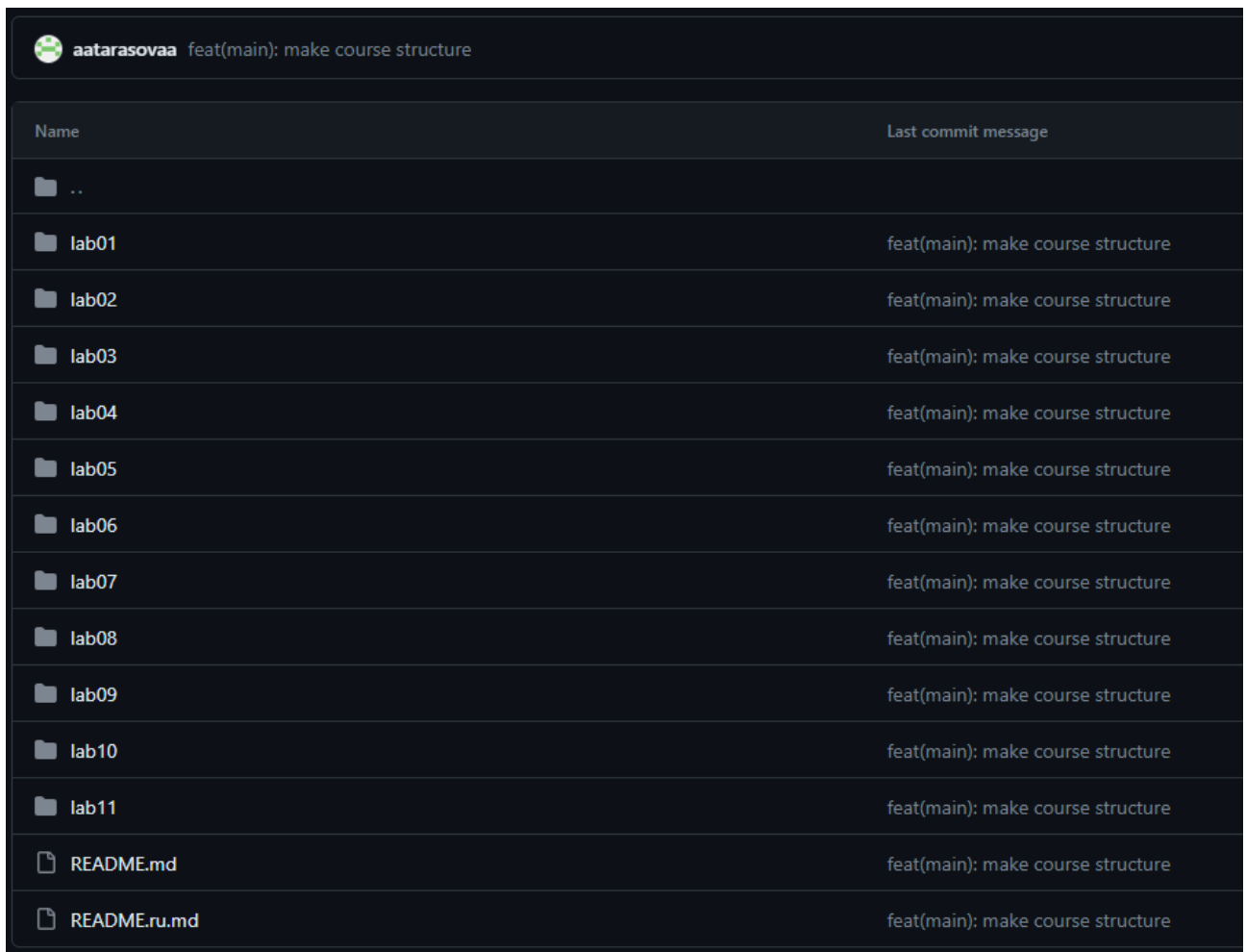
```
aatarasova@aatarasova-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/a  
rch-pc$ git push  
Перечисление объектов: 37, готово.  
Подсчет объектов: 100% (37/37), готово.  
Сжатие объектов: 100% (29/29), готово.  
Запись объектов: 100% (35/35), 342.13 КиБ | 1.17 Миб/с, готово.  
Всего 35 (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:aatarasovaa/study_2023-2024_arhpc.git  
08ed52c..fd4c55f master -> master
```

Рис. 4.21. Выгрузка изменений на сервер.

Проверяю правильность создания иерархии рабочего пространства в локальном репозитории (рис.4.22) и на странице github (рис.4.23).

```
aatarasova@aatarasova-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/a  
rch-pc$ ls  
CHANGELOG.md  COURSE  LICENSE  prepare  README.en.md  README.md  
config        labs    Makefile  presentation  README.git-flow.md  template  
aatarasova@aatarasova-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/a  
rch-pc$ ls labs  
lab01  lab03  lab05  lab07  lab09  lab11  README.ru.md  
lab02  lab04  lab06  lab08  lab10  README.md
```

Рис.4.22. Локальный репозиторий.



Name	Last commit message
..	
lab01	feat(main): make course structure
lab02	feat(main): make course structure
lab03	feat(main): make course structure
lab04	feat(main): make course structure
lab05	feat(main): make course structure
lab06	feat(main): make course structure
lab07	feat(main): make course structure
lab08	feat(main): make course structure
lab09	feat(main): make course structure
lab10	feat(main): make course structure
lab11	feat(main): make course structure
README.md	feat(main): make course structure
README.ru.md	feat(main): make course structure

Рис. 4.25. Страница репозитория на GitHub.

## 4.5 Выполнение заданий для самостоятельной работы.

1. Перехожу в директорию labs/lab03/report с помощью утилиты cd.

Создаю в каталоге файл для отчета по третьей лабораторной работе с помощью утилиты touch (рис. 4.26).

```
aatarasova@aatarasova-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/аrch-pc$ cd labs/lab02/report
aatarasova@aatarasova-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/аrch-pc/labs/lab02/report$ touch Л02_Тарасова_отчет
```

Рис. 4.26: Создание файла.

Оформить отчет я смогу в текстовом процессоре LibreOffice Writer (рис. 4.27).

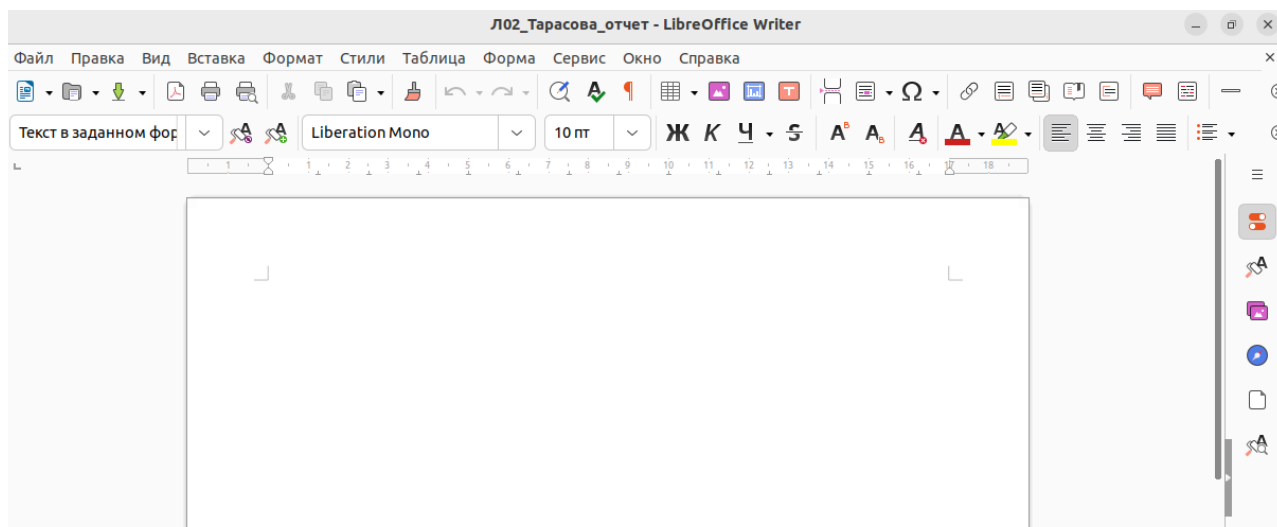


Рис. 4.27. Работа с отчетом в текстовом процессоре.

2.



## **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. Архитектура ЭВМ.
2. Git — gitattributes Документация.