

Отчет по лабораторной работе №2

Дисциплина - архитектура компьютера

Жернаков Данила Иванович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Настройка GitHub	9
4.2	Базовая настройка Git	10
4.3	Создание SSH-ключа	11
4.4	Создание рабочего пространства и репозитория курса на основе шаблона	12
4.5	Создание репозитория курса на основе шаблона	12
4.6	Настройка каталога курса	15
4.7	Выполнение заданий для самостоятельной работы	16
5	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	Заполнение данных учетной записи GitHub	9
4.2	Аккаунт GitHub	10
4.3	Предварительная настройка Git	11
4.4	Генерация SSH-ключа	11
4.5	Окно SSH and GPG keys	12
4.6	Создание рабочего пространства	12
4.7	Страница шаблона для репозитория	13
4.8	Окно создания репозитория	13
4.9	Созданный репозиторий	14
4.10	Клонирование репозитория	14
4.11	Окно с ссылкой для копирования репозитория	15
4.12	Создание каталогов	15
4.13	Добавление и сохранение изменений на сервере	15
4.14	Выгрузка изменений на сервер	16
4.15	Выгрузка изменений на сервер	16
4.16	Создание файла	16
4.17	Перемещение отчета	17
4.18	Добавление файла на сервер	17
4.19	Добавление файла на сервер	17
4.20	Подкаталоги и файлы в репозитории	18
4.21	Отправка в центральный репозиторий сохраненных изменений	18
4.22	Страница каталога в репозитории	18
4.23	Страница последних изменений в репозитории	19

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

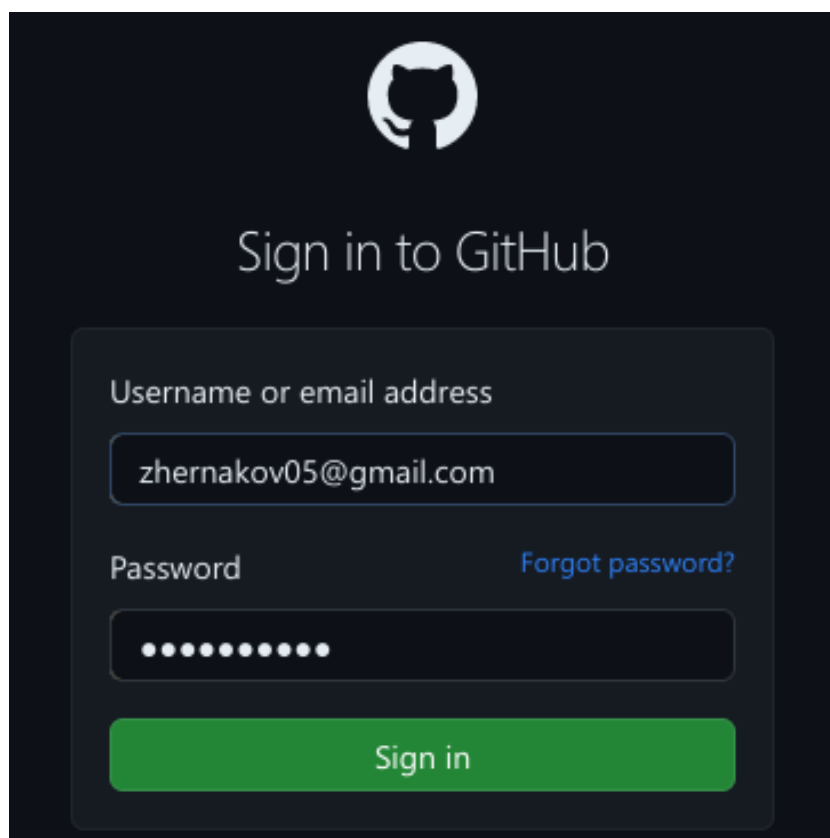
Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля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

4 Выполнение лабораторной работы

4.1 Настройка GitHub

Захожу в свою учетную запись GitHub (рис. 4.1) (рис. 4.2).



Sign in to GitHub

Username or email address

zhernakov05@gmail.com

Password [Forgot password?](#)

Sign in

Рис. 4.1: Заполнение данных учетной записи GitHub

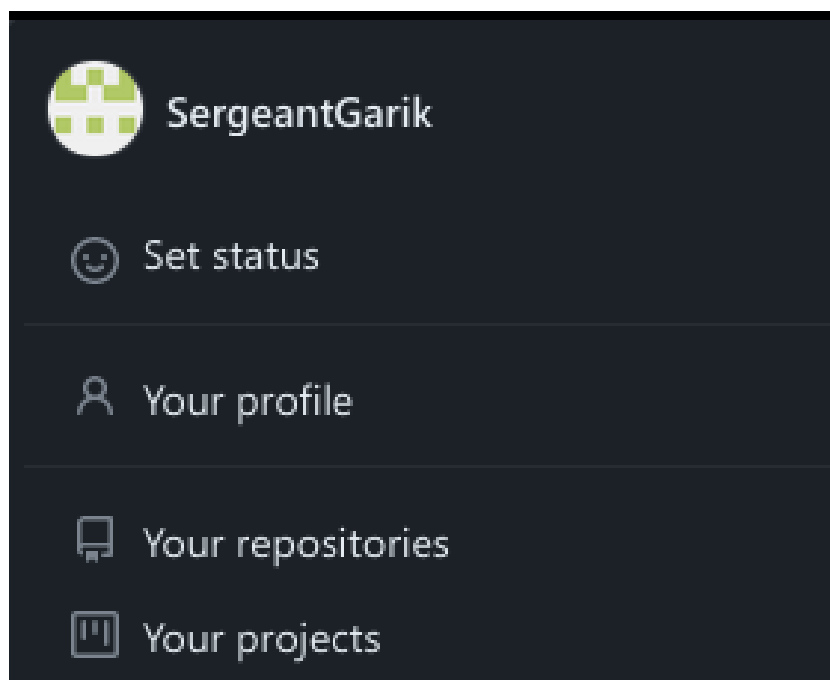


Рис. 4.2: Аккаунт GitHub

4.2 Базовая настройка Git

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. Настраиваю utf-8 в выводе сообщений git для корректного отображения символов. Задаю имя «master» для начальной ветки. Задаю параметр `autocrlf` со значением `input`, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах. CR и LF – это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах. Задаю параметр `safecrlf` со значением `warn`, так Git будет проверять преобразование на обратимость. При значении `warn` Git только выведет предупреждение, но будет принимать необратимые конвертации. (рис. 4.3).

```
dizhernakov@dk4n63 ~ $ git config --global user.name "Danila Zhernakov"
gitdizhernakov@dk4n63 ~ $ git config --global user.email "zhernakov05@gmail.com"
dizhernakov@dk4n63 ~ $ git config --global core.quotepath false
dizhernakov@dk4n63 ~ $ git config --global init.defaultBranch master
dizhernakov@dk4n63 ~ $ git config --global core.autocrlf input
dizhernakov@dk4n63 ~ $ git config --global core.safecrlf warn
```

Рис. 4.3: Предварительная настройка Git

4.3 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 (рис. 4.4). Ключ автоматически сохранится в каталоге `~/.ssh/`.

```
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/d/i/dizhernakov/.ssh/id_rsa):
Created directory '/afs/.dk.sci.pfu.edu.ru/home/d/i/dizhernakov/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/d/i/dizhernakov/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/d/i/dizhernakov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:uuT9B5gd+eXoRqX0m+qLReCFNNRcDgQp7w1ld+lk7Mw Danila Zhernakov zhernakov05@gmail.com
The key's randomart image is:
+---[RSA 3072]---+
| .=*O... |
| ...O=O. * |
| ooo.o.O |
| .ooo E |
| .S*.o * |
| .+. + = |
| o . * |
| o o o * |
| o oo=O. |
+---[SHA256]-----+
```

Рис. 4.4: Генерация SSH-ключа

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» (рис. 4.5).

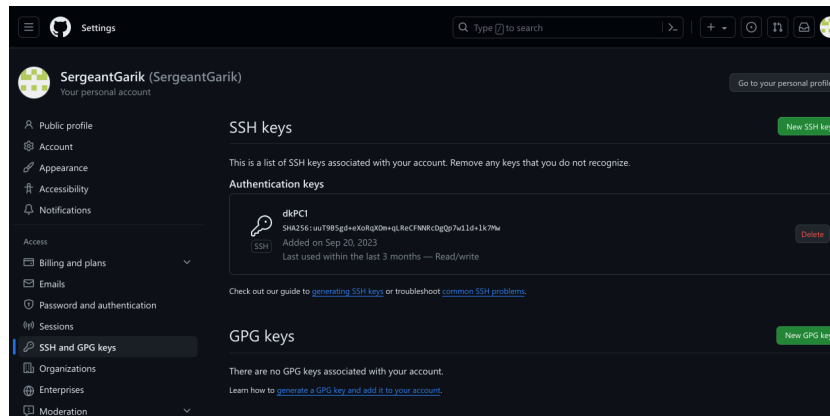


Рис. 4.5: Окно SSH and GPG keys

Ключ уже был добавлен и сейчас отображается в этом окне

4.4 Создание рабочего пространства и репозитория курса на основе шаблона

Закрываю браузер, открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты `mkdir`, благодаря ключу `-p` создаю все директории после домашней `~/work/study/2023-2024/“Архитектура компьютера”` рекурсивно. Через терминал перехожу в созданный каталог курса с помощью утилиты `cd` (рис. 4.6).

```
dizhernakov@dk4n63 ~ $ mkdir -p ~/work/study/2023-2024/"Архитектура компьютера"
dizhernakov@dk4n63 ~ $ cd ~/work/study/2023-2024/Архитектура\ компьютера/
dizhernakov@dk4n63 ~/work/study/2023-2024/Архитектура компьютера $ git clone --
```

Рис. 4.6: Создание рабочего пространства

4.5 Создание репозитория курса на основе шаблона

В браузере перехожу на страницу репозитория с шаблоном курса по адресу <https://github.com/yamadharma/course-directory-student-template>. Далее выбираю

«Use this template», чтобы использовать этот шаблон для своего репозитория (рис. 4.7).

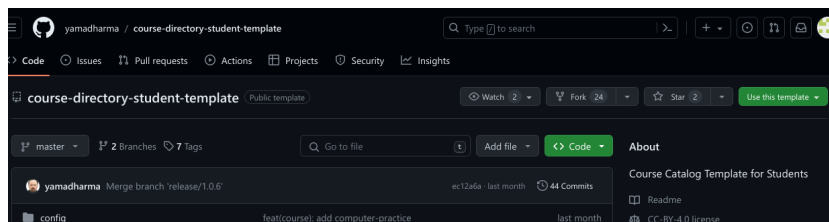


Рис. 4.7: Страница шаблона для репозитория

В открывшемся окне задаю имя репозитория (Repository name): study_2023–2024_arh-
pc и создаю репозиторий, нажимаю на кнопку «Create repository from template»
(рис. 4.8).

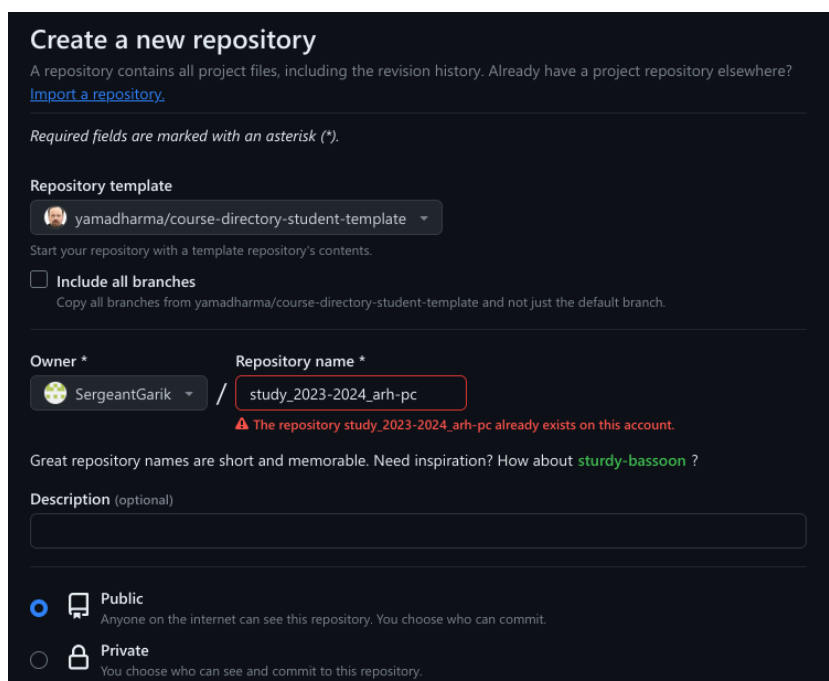


Рис. 4.8: Окно создания репозитория

Репозиторий создан (рис. 4.9).

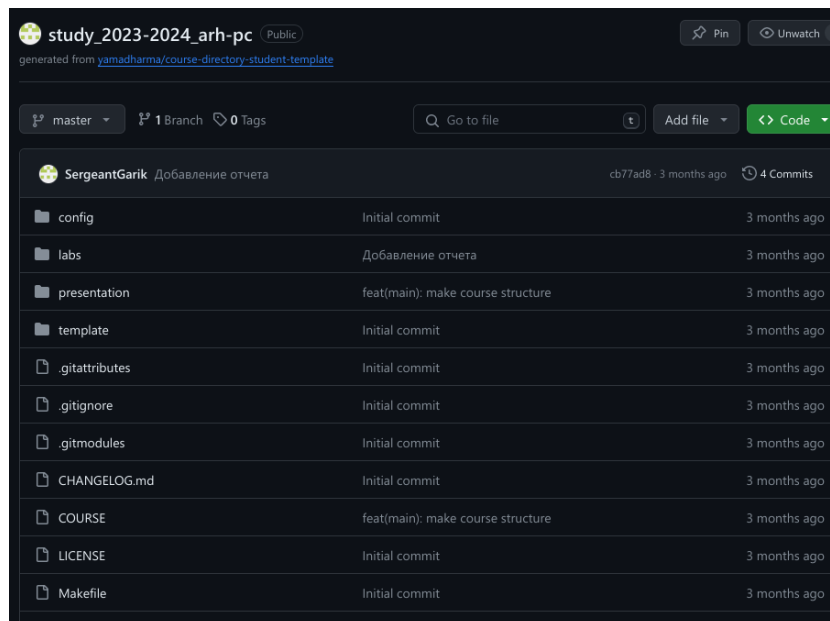


Рис. 4.9: Созданный репозиторий

Клонирую созданный репозиторий с помощью команды `git clone --recursive git@github.com:/study_2022-2023_arh-pc.git arch-pc` (рис. 4.10).

```
dizhernakov@dk4n63 ~ $ mkdir -p ~/work/study/2023-2024/Архитектура компьютера
dizhernakov@dk4n63 ~ $ cd ~/work/study/2023-2024/Архитектура компьютера/
dizhernakov@dk4n63 ~/work/study/2023-2024/Архитектура компьютера $ git clone --recursive git@github.com:SergeantGarik/study_2023-2024_arh-pc.git arch-pc
Клонирование в «arch-pc»...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
```

Рис. 4.10: Клонирование репозитория

Копирую ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. 4.11).

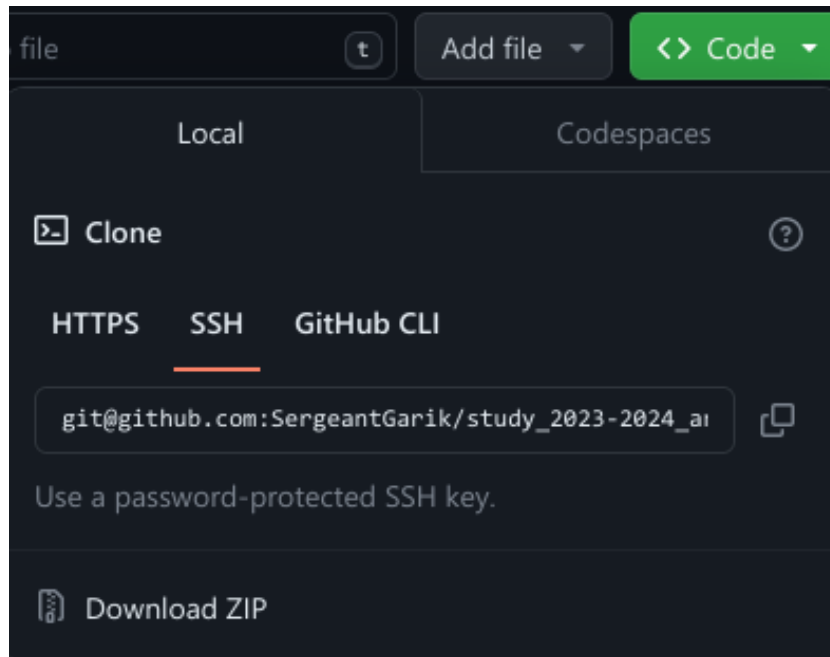


Рис. 4.11: Окно с ссылкой для копирования репозитория

4.6 Настройка каталога курса

Перехожу в каталог arch-pc и создаю необходимые каталоги (рис. 4.12).

```
Клонирование в ~/afs/dk.sci.pfu.edu.ru/home/d/i/dizhernakov/work/study/2023-2024/Архитектура компьютера/arch-pc $ cd ..
~/work/study/2023-2024/Архитектура компьютера/arch-pc $ echo arch-pc > COURSE
dizhernakov@dk4n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ make
```

Рис. 4.12: 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. 4.13).

```
dizhernakov@dk4n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs $ cd ..
dizhernakov@dk4n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git add .
dizhernakov@dk4n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git commit -am 'feat(main): make course structure'
[master 95f12a8] 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
```

Рис. 4.13: Добавление и сохранение изменений на сервере

Отправляю все на сервер с помощью push (рис. 4.14).

```
dizhernakov@dk4n63 ~/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 6 потоков
Сжатие объектов: 100% (29/29), готово.
Запись объектов: 100% (35/35), 342.14 КиБ | 2.78 МБ/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:SergeantGarik/study_2023-2024_arch-pc.git
   ac181e5..95f12a0  master -> master
```

Рис. 4.14: Выгрузка изменений на сервер

Проверяю правильность выполнения работы сначала на самом сайте GitHub (рис. 4.15).

Name	Last commit message	Last commit date
..		
lab01	Добавление отчета	3 months ago
lab02	feat(main): make course structure	3 months ago
lab03	feat(main): make course structure	3 months ago
lab04	feat(main): make course structure	3 months ago
lab05	feat(main): make course structure	3 months ago
lab06	feat(main): make course structure	3 months ago
lab07	feat(main): make course structure	3 months ago
lab08	feat(main): make course structure	3 months ago
lab09	feat(main): make course structure	3 months ago
lab10	feat(main): make course structure	3 months ago
lab11	feat(main): make course structure	3 months ago
README.md	feat(main): make course structure	3 months ago

Рис. 4.15: Выгрузка изменений на сервер

4.7 Выполнение заданий для самостоятельной работы

1. Создаю в каталоге labs/lab02/report файл для отчета по второй лабораторной работе с помощью утилиты touch (рис. 4.16).

```
dizhernakov@dk4n60 ~ $ touch work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/lab02/
report/Л02_Хернаков_отчет
dizhernakov@dk4n60 ~ $
```

Рис. 4.16: Создание файла

2. Перемещаю доклад для первой лабораторной работы из папки «Документы» в соответствующую директорию в новом рабочем пространстве при помощи утилиты mv (рис. 4.17).

```
dizhernakov@dk4n60 ~$ mv Документы/Л01_Жернаков_отчет.pdf work/study/2023-2024/Архитектура\
компьютера/arch-pc/labs/lab01/report/
dizhernakov@dk4n60 ~$
```

Рис. 4.17: Перемещение отчета

3. Добавляю с помощью команды git add в коммит отчет по первой лабораторной работе. (рис. 4.18).

```
dizhernakov@dk4n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report $
git add Л01_Жернаков_отчет.pdf
dizhernakov@dk4n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report $
```

Рис. 4.18: Добавление файла на сервер

Перехожу в директорию, в которой находится отчет по второй лабораторной работе с помощью cd и добавляю файл Л02_Жернаков_отчет (рис. 4.19).

```
dizhernakov@dk4n60 ~$ cd work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/lab02/rep
ort/
dizhernakov@dk4n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report $
git add Л02_Жернаков_отчет.pdf
fatal: спецификатор пути «Л02_Жернаков_отчет.pdf» не соответствует ни одному файлу
dizhernakov@dk4n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report $
git add Л02_Жернаков_отчет.pdf
fatal: спецификатор пути «Л02_Жернаков_отчет.pdf» не соответствует ни одному файлу
dizhernakov@dk4n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report $
cp Л02_Жернаков_отчет.odt Л02_Жернаков_отчет.pdf
dizhernakov@dk4n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report $
git add Л02_Жернаков_отчет.pdf
dizhernakov@dk4n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report $
```

Рис. 4.19: Добавление файла на сервер

Сохраняю изменения на сервере командой git commit -m "...", поясняя, что добавил файлы. (рис. 4.20).

```
dizhernakov@dk4n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report $
git commit -m "Добавление отчета"
[master 09d7b11] Добавление отчета
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 labs/lab01/report/Л01_Хернаков_отчет.pdf
create mode 100644 labs/lab02/report/Л02_Хернаков_отчет.pdf
dizhernakov@dk4n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report $
```

Рис. 4.20: Подкаталоги и файлы в репозитории

Отправляю в центральный репозиторий сохраненные изменения командой `git push -f origin master` (рис. 4.21).

```
dizhernakov@dk4n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report $
git push origin master
Перечисление объектов: 14, готово.
Подсчет объектов: 100% (14/14), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (8/8), готово.
Запись объектов: 100% (8/8), 1.28 МиБ | 2.03 МиБ/с, готово.
Всего 8 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To github.com:SergeantGarik/study_2023-2024_arh-pc.git
cb77ad8..09d7b11 master -> master
```

Рис. 4.21: Отправка в центральный репозиторий сохраненных изменений

Проверяю на сайте GitHub правильность выполнения заданий. Вижу, что пояснение к совершенным действиям отображается (рис. 4.22).

Name	Last commit message	Last commit date
..		
lab01	Добавление отчета	2 minutes ago
lab02	Добавление отчета	2 minutes ago

Рис. 4.22: Страница каталога в репозитории

При просмотре изменений так же вижу, что были добавлены файлы с отчетами по лабораторным работам (рис. 4.23).

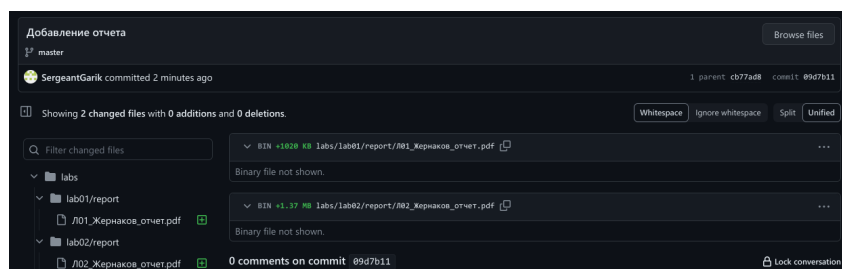


Рис. 4.23: Страница последних изменений в репозитории

5 Выводы

При выполнении данной лабораторной работы я изучил 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