Отчёт по лабораторной работе №2

Дисциплина: Архитектура Компьютера

Дарина Андреевна Куокконен

Содержание

# 1 Цель работы

Целью данной работы является приобретение практического опыта работы с системой git, а также изучение принципов и применения контроля версий.

# 2 Задание

1. Настройка GitHub.  
2. Базовая настройка Git.  
3. Создание SSH-ключа.  
4. Создание рабочего пространства.  
5. Создание репозитория курса на основе шаблона.  
6. Настройка каталога курса.  
7. Выполнение задания 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 (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 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. 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

1. Настройка GitHub

Создаю учетную запись на сайте GitHub, ввожу нужные данные учетной записи (рис. [1](#fig:1))

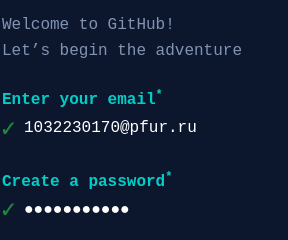


Figure 1: Создание учетной записи GitHub

Figure 2: Создание учетной записи GitHub

Figure 2: Создание учетной записи GitHub

Создание аккаунта (рис. [3](#fig:3))



Figure 3: Создание аккаунта GitHub

1. Базовая настройка Git

Открываю виртуальную машину, затем запускаю терминал, 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 этой команде свою электронную почту, настраиваю utf-8 в выводе сообщений git для корректного отображения символов (рис. [4](#fig:4))

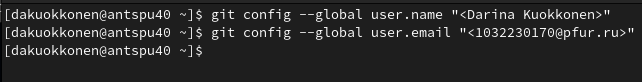


Figure 4: Предварительная конфигурация git

Настраиваю urf-8 в выводе сообщений git для корректного отображения символов(рис. [5](#fig:5))

Figure 5: Настройка кодировки

Figure 5: Настройка кодировки

Задаю имя “master” для начальной ветки(рис. [6](#fig:6))

Figure 6: Создание имени для начальной ветки

Figure 6: Создание имени для начальной ветки

Задаю имя начальной ветке, а также параметры autocrlf и safecrlf, причем параметр autocrlf дополняем значением input, для конвертации символов разрыва строки в текстовых файлах (CRLF и LF) только при коммитах. Параметру safecrlf задаю значение warn, так Git будет проверять преобразование на обратимость, и при данном значении будет выведено только предупреждение, а необратимые конвертации будут приняты (рис. [7](#fig:7))

Figure 7: Создание имени для начальной ветки, присваивание параметров

Figure 7: Создание имени для начальной ветки, присваивание параметров

Задаю параметр autocrlf со значением input, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах (рис. 4.6). CR и LF - это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах. (рис. [8](#fig:8))

Figure 8: Параметр autocrlf

Figure 8: Параметр autocrlf

Задаю параметр safecrIf со значением warn, так Git будет проверять преобразование на обратимость. При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.(рис. [9](#fig:9))

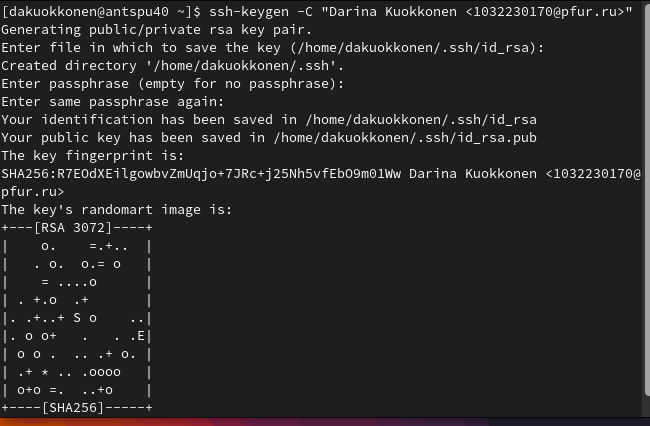


Figure 9: Параметр safecrlf

1. Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый). Для этого ввожу команду ssh-keygen -C “Имя Фамилия, work@email”, указывая своё имя и электронную почту. Ключ автоматически сохранится в каталоге ~/.ssh/. (рис. [10](#fig:10))

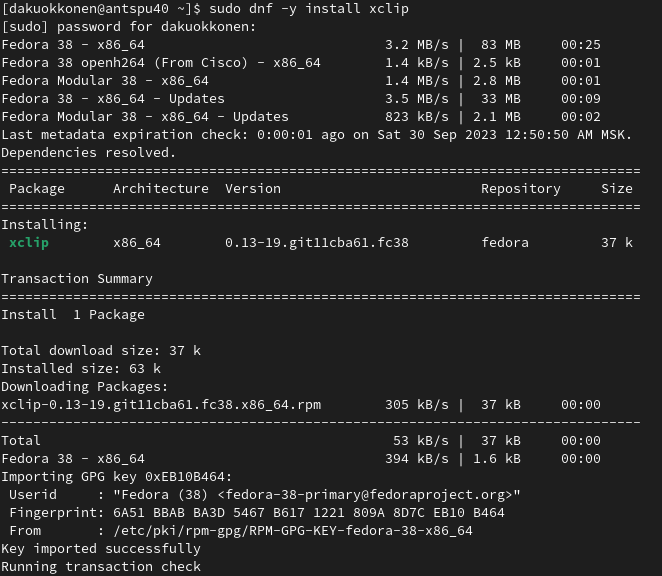


Figure 10: Генерация SSH-ключа

Устанавливаю утилиту xclip, позволяющую копировать любой текст через терминал. Использую команду «dnf install» с ключом -y от имени суперпользователя, ввожу в начале «sudo» (рис. ??) Установка утилиты xclip

Копирую открытый ключ из директории, в которой он был сохранен, с помощью утилиты xclip (рис. [11](#fig:12))

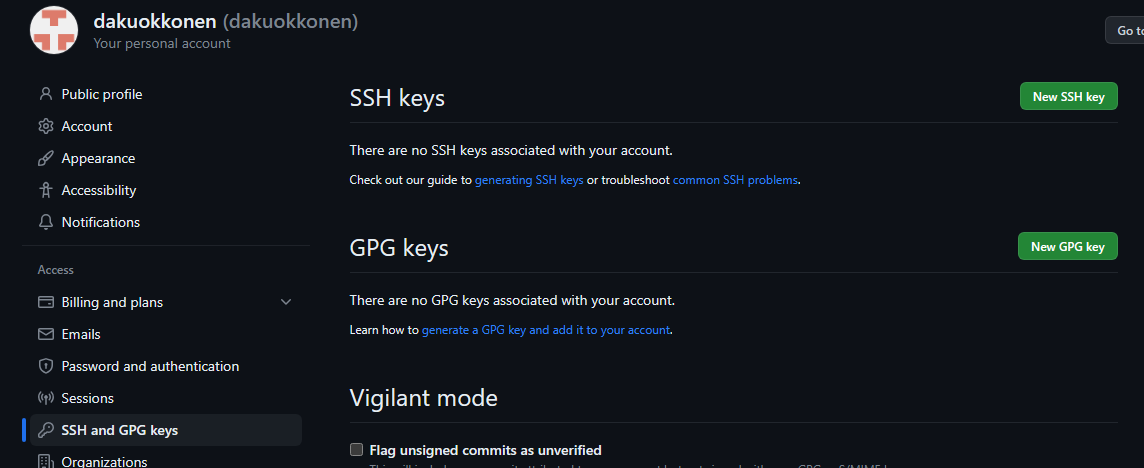


Figure 11: Копирование ключа

Захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key». Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы завершить добавление ключа. Вставляю скопированный ключ в поле “Key”. В поле Title указываю имя для ключа.(рис. [12](#fig:13))

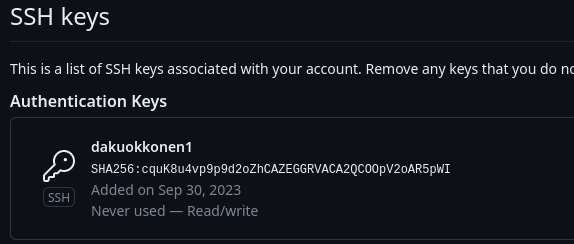


Figure 12: Добавление ключа

1. Создание рабочего пространства

Закрываю браузер, открываю терминал. Создаю директорию, рабочее пространство с помощью утилиты mkdir. С помощью ключа -p создаю рекурсивно все директории после домашней ~/work/study/2023-2024/“Архитектура Компьютера”. Далее проверяю с помощью ls, действительно ли были созданы необходимые мне каталоги (рис. [13](#fig:14))

Figure 13: Создание рабочего пространства

Figure 13: Создание рабочего пространства

1. 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. [14](#fig:15))

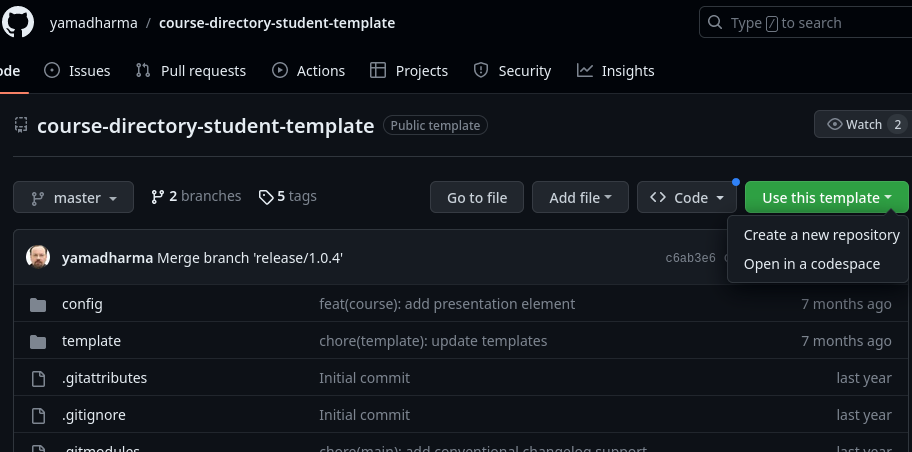


Figure 14: Страница репозитория с шаблоном

Далее, задаю имя репозитория и создаю репозиторий.(рис. [15](#fig:16))



Figure 15: Окно создания репозитория

Ждем создания репозитория, затем он откроется (рис. [16](#fig:17))

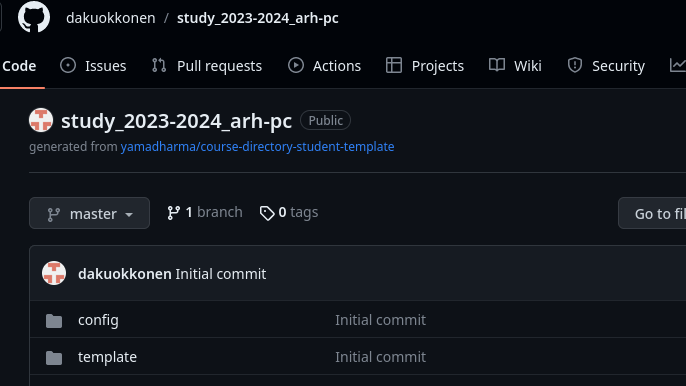


Figure 16: Созданный репозиторий

Далее через терминал перехожу в созданный каталог курса с помощью утилиты cd.(рис. [17](#fig:18))

Figure 17: Перемещение между директориями

Figure 17: Перемещение между директориями

Клонирую созданный репозиторий с помощью команды: «git clone –recursive git@github.com:/study\_2023–2024\_arh-pc.git arch-pc» (рис. [18](#fig:19))

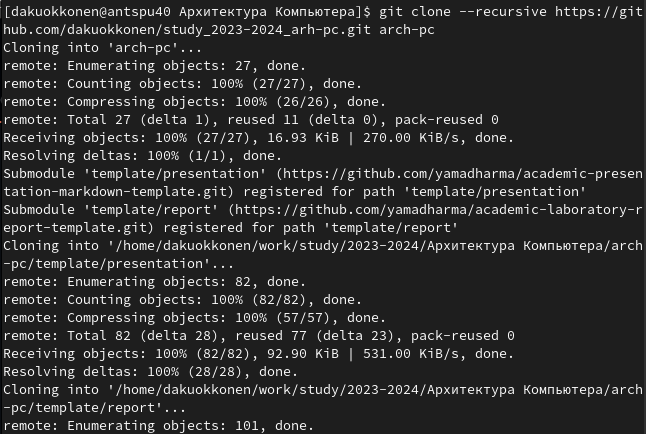


Figure 18: Клонирование репозитория

Ссылку для копирования можно взять на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. [19](#fig:20))

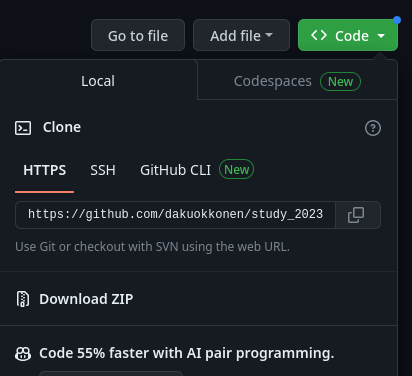


Figure 19: Окно с ссылкой на копирование репозитория

1. Настройка каталога курса

Перехожу в каталог arch-pc с помощью утилиты cd. (рис. [20](#fig:21))

Figure 20: Перемещение между директориями

Figure 20: Перемещение между директориями

Затем удаляю лишние файлы с помощью утилиты rm.(рис. [21](#fig:22))

Figure 21: удаление файлов

Figure 21: удаление файлов

Далее, создаю необходимые каталоги (рис. [22](#fig:23))

Figure 22: Создание каталогов

Figure 22: Создание каталогов

Отправляю созданные каталоги с локального репозитория на сервер: добавляю все созданные каталоги с помощью «git add», комментирую и сохраняю изменения на сервере с помощью «git commit» (рис. [23](#fig:24))

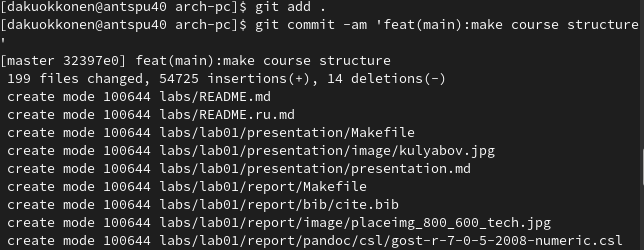


Figure 23: Добавление и сохранение изменений на сервере

Отправляю все на сервер с помощью «push» (рис. [24](#fig:25))

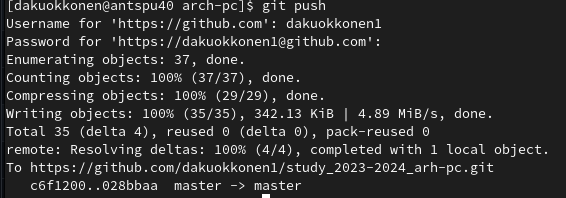


Figure 24: Выгрузка изменений на сервер

Проверяю правильность выполнения работы на самом сайте GitHub (рис. [25](#fig:26))

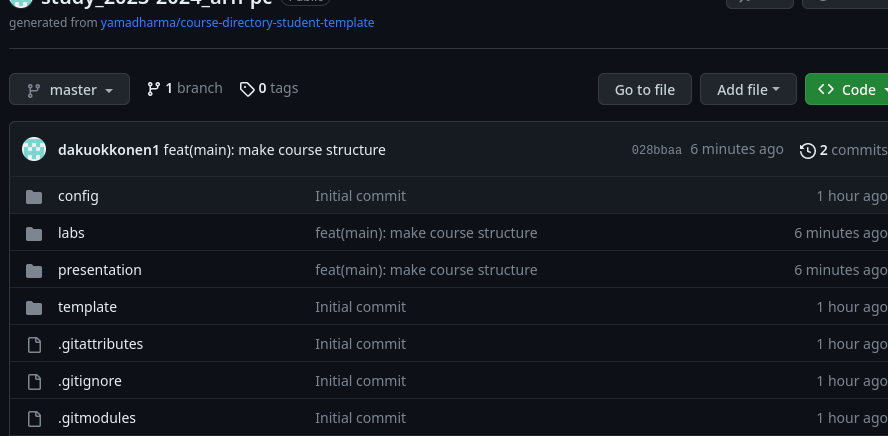


Figure 25: Проверка результата на странице репозитория

1. Выполнение заданий для самостоятельной работы

1)Перехожу в директорию labs/lab02/report с помощью утилиты cd. Создаю в каталоге файл для отчета по третьей лабораторной работе с помощью утилиты touch. (рис. [26](#fig:27))

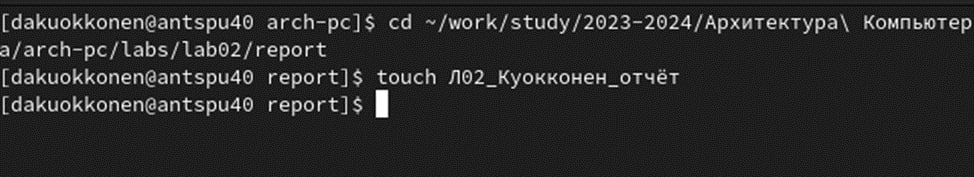


Figure 26: Перемещение по директориям, создание файла

Оформить отчет я смогу в текстовом процессоре LibreOffice Writer, найдя его в меню приложений. После открытия текстового процессора открываю в нем созданный файл. Теперь можно создать в нем отчет (рис. [27](#fig:28))

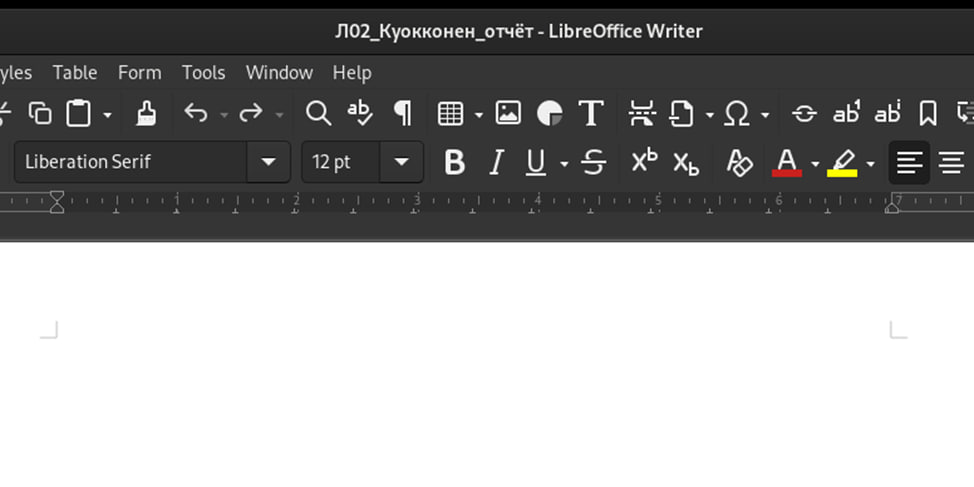


Figure 27: Работа с отчётом в текстовом редакторе

2)Проверяю местонахождение файла с отчетом по первой лабораторной работе. (рис. [28](#fig:29))

Figure 28: Проверка местонахождения файла

Figure 28: Проверка местонахождения файла

Использую утилиту “cd”, чтобы попасть в подкаталог lab02/report, в котором находится отчет по первой лабораторной работе. (рис. [29](#fig:30))

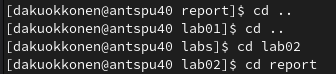


Figure 29: Перемещение по директориям

Добавляю файл Л01\_Куокконен\_отчёт(рис. [30](#fig:31))

Figure 30: Добавление файла на сервер

Figure 30: Добавление файла на сервер

То же самое делаю для отчета по второй лабораторной работе, перехожу в директорию labs/lab02/report с помощью утилиты cd, добавляю с помощью git add нужный мне файл, сохраняю изменения с помощью git commit(рис. [31](#fig:32))

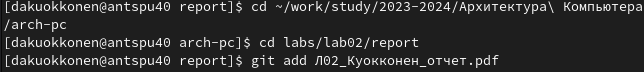


Figure 31: Подкаталоги и файлы в репозитории

Отправляю данный файл в центральный репозиторий, с помощью команды git push -f origin master.(рис. [32](#fig:33))

Figure 32: Отправка файла

Figure 32: Отправка файла

Проверяю на сайте GitHub правильность выполнения заданий.(рис. [33](#fig:34)),(рис. [34](#fig:35))

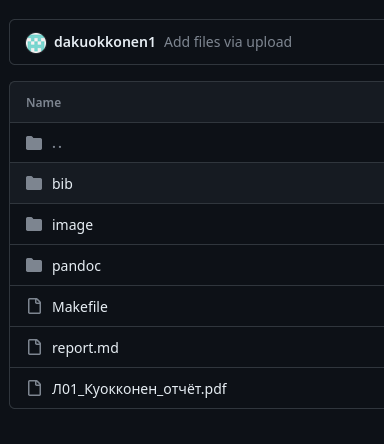


Figure 33: Проверка результата на страницах каталога репозитория

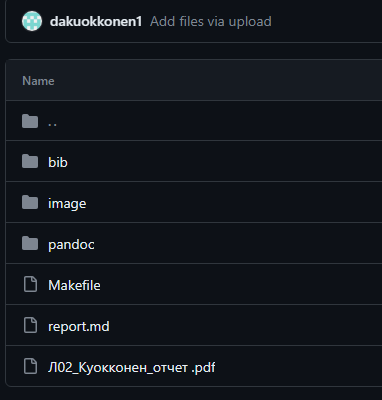


Figure 34: Проверка результата на страницах каталога репозитория

# 5 Выводы

Я приобрела практический опыт работы с системой Git, изучила принципы и применение контроля версий.

# Список литературы

1. Архитектура ЭВМ (rudn.ru)  
2. Инструкция по использованию Git