Архитектура компьютеров и операционные системы | Операционные системы

Лабораторная работа № 3. Markdown

Акрур Имад НКАбд-06-24

Содержание

# 1 Цель работы

* Научиться оформлять отчёты с помощью легковесного языка разметки **Markdown**.

# 2 Задание

* Сделать отчёт по предыдущей лабораторной работе в формате **Markdown**.
* В качестве отчёта нужно предоставить отчёты в **3 форматах**: **pdf**, **docx** и **md** (в **архиве**, поскольку он должен содержать **скриншоты**, **Makefile** и т.д.)

# 3 Теоретическое введение :

### 3.0.1 Контрольные вопросы для самопроверки

1. **Что такое Markdown?**  
   Markdown — это легкий язык разметки, который позволяет форматировать текст с помощью простых символов, делая его легко читаемым и редактируемым.
2. **Как в Markdown задается начертание шрифтов?**
   * *Курсив*: обрамление текста в одиночные звездочки или подчеркивания, например, \*курсив\* или \_курсив\_.
   * **Жирный**: обрамление текста в двойные звездочки или подчеркивания, например, \*\*жирный\*\* или \_\_жирный\_\_.
   * ***Жирный курсив***: обрамление текста в три звездочки, например, \*\*\*жирный курсив\*\*\*.
3. **Как в Markdown оформляются списки?**
   * **Ненумерованный список**: используется звездочка, плюс или дефис:
   * - Пункт 1  
     - Пункт 2
   * **Нумерованный список**: используются числа с точками:
   * 1. Первый пункт  
     2. Второй пункт
4. **Как в Markdown оформляются изображения и ссылки на них?**
   * **Изображение**: ![Alt text](url\_изображения)
   * **Ссылка**: [Текст ссылки](url\_ссылки)
5. **Как в Markdown оформляются математические формулы и ссылки на них?**
   * Для отображения формул используйте знаки доллара:
     + Внутри строки: $E=mc^2$
     + На отдельной строке:
     + $$E=mc^2$$

# 4 Выполнение лабораторной работы

## 4.1 Цель второй лабораторной работы:

* Изучить идеологию и применение средств контроля версий.
* Освоить умения по работе с git.

## 4.2 Теоретическое введение:

### 4.2.1 Контрольные вопросы:

**1. Что такое системы контроля версий (VCS) и для чего они предназначаются?**

Системы контроля версий (VCS) — это инструменты для управления изменениями в коде и файлах проекта, позволяющие отслеживать историю изменений, работать с разными версиями и сотрудничать в команде.

**2. Объясните понятия VCS: хранилище, commit, история, рабочая копия**

- Хранилище repositoryместо, где хранятся все версии проекта.

- Commit зафиксированные изменения с комментарием, представляющие версию проекта.

- История после довательность всех commit

- Рабочая копия working copyлокальная версия проекта, с которой работает разработчик.

**3. Чем отличаются централизованные и децентрализованные VCS? Примеры.**

- Централизованные VCS (например, SVN): одно центральное хранилище, доступное для всех.

- Децентрализованные VCS (например, Git): каждый пользователь имеет своё полное хранилище, с возможностью синхронизации.

**4. Действия с VCS при единоличной работе с хранилищем.**

- Инициализация репозитория.

- Добавление файлов и создание commit

- Проверка состояния (git status), просмотр истории (git log).

**5. Порядок работы с общим хранилищем VCS.**

- Клонирование репозитория.

- Создание ветки, внесение изменений.

- Commit изменений.

- Слияние с основной веткой через pull request.

**6. Основные задачи, решаемые Git.**

Управление версиями, создание веток, слияние изменений, разрешение конфликтов, работа с удалёнными репозиториями.

**7. Команды Git: краткая характеристика.**

- `git init`: создание репозитория.

- `git clone`: клонирование удалённого репозитория.

- `git add`: добавление файлов для отслеживания.

- `git commit`: фиксация изменений.

- `git push`: отправка изменений в удалённый репозиторий.

- `git pull`: получение изменений из удалённого репозитория.

**8. Примеры работы с локальными и удалёнными репозиториями.**

- Локальные : инициализация репозитория, создание commit , проверка состояния.

- Удалённые: клонирование, отправка изменений (git push), получение обновлений (git pull).

### 4.2.2 Выполнение лабораторной работы:

#### 4.2.2.1 Создание учетной записи на GitHub

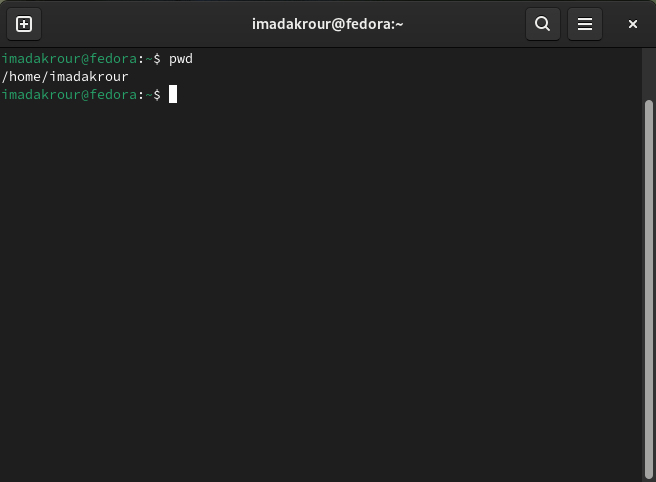


Рис. 1: рисунок 1

*в моем случае учетная запись уже существует, поэтому этот шаг был пропущен.*

#### 4.2.2.2 Базовая настройка Git**:**

1. **Указание имени пользователя и email для создания коммитов в репозиториях. Эти данные будут добавляться к каждому коммиту, чтобы идентифицировать автора изменений.:**

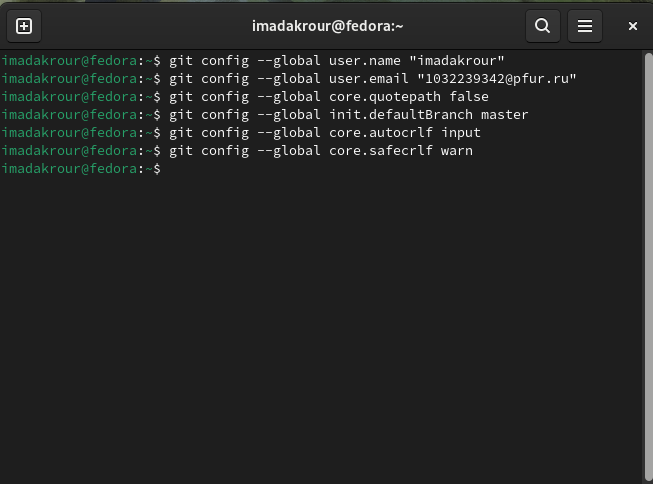


Рис. 2: рисунок 2

*Эта последовательность команд настраивает имя пользователя, email, кодировку UTF-8, задает начальную ветку master, устанавливает обработку концов строк (autocrlf), и включает предупреждения о несоответствиях концов строк (safecrlf). Выполнение всех команд в одной строке позволяет быстрее закончить настройку и сразу перейти к работе с Git. Все параметры будут применены глобально для всех будущих репозиториев.*

#### 4.2.2.3 Создание SSH-ключа

**Описание выполняемого задания:**

Для безопасного подключения к репозиториям на GitHub, нужно сгенерировать SSH-ключи (публичный и приватный). Это позволит вам работать с репозиториями, не вводя каждый раз логин и пароль.

1. **Генерация SSH-ключа:**

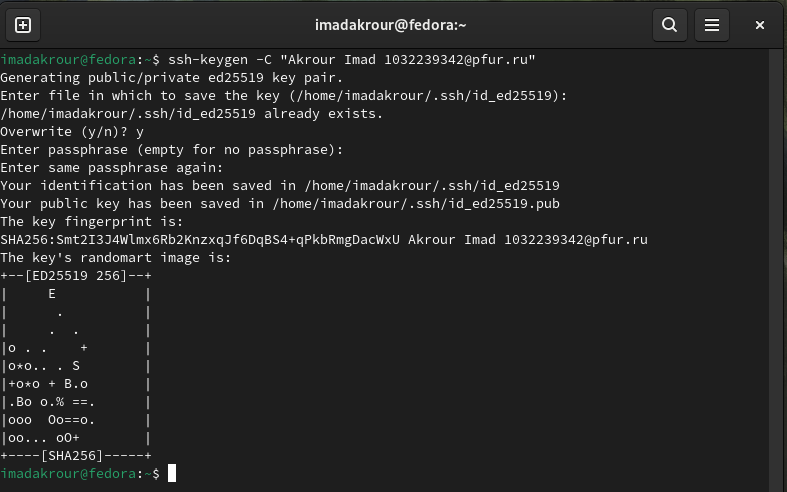


Рис. 3: рисунок 3

*Каталог ~/.ssh/ — это стандартное место для хранения SSH-ключей. Не изменяйте путь, если вы не хотите использовать другое место для хранения.*

1. **Копирование публичного ключа в буфер обмена:**

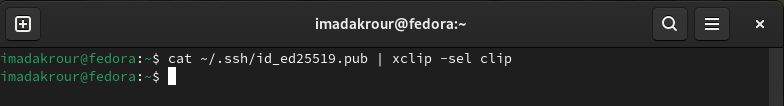


Рис. 4: рисунок 4

*Команда cat ~/.ssh/id\_rsa.pub выводит содержимое публичного ключа, а команда xclip -sel clip копирует это содержимое в буфер обмена*

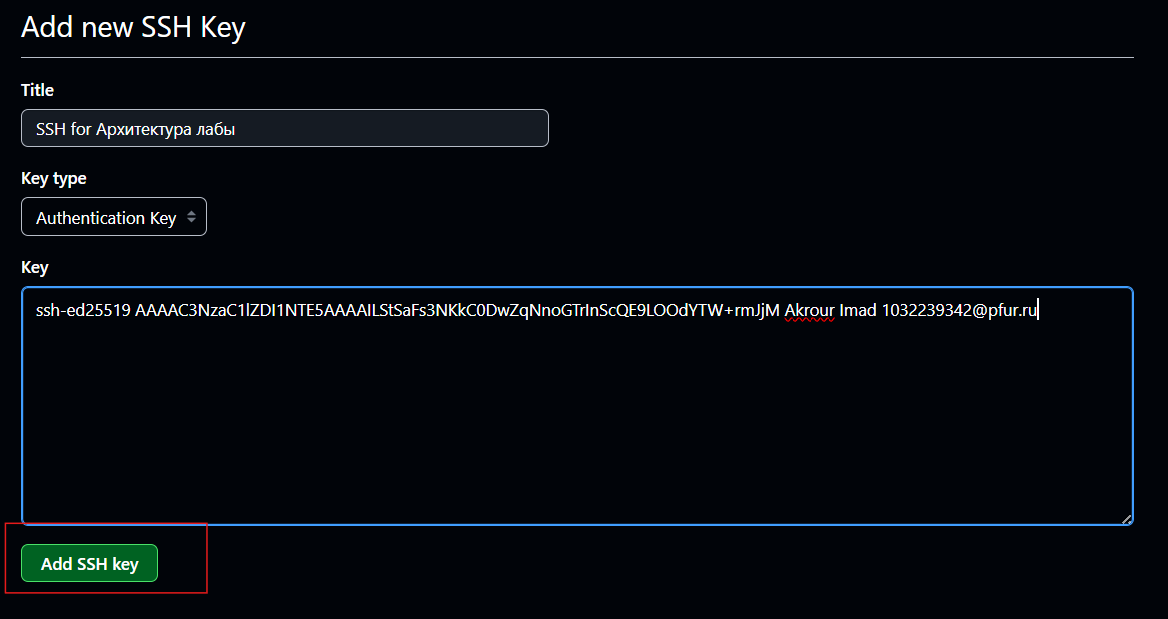


Рис. 5: рисунок 5

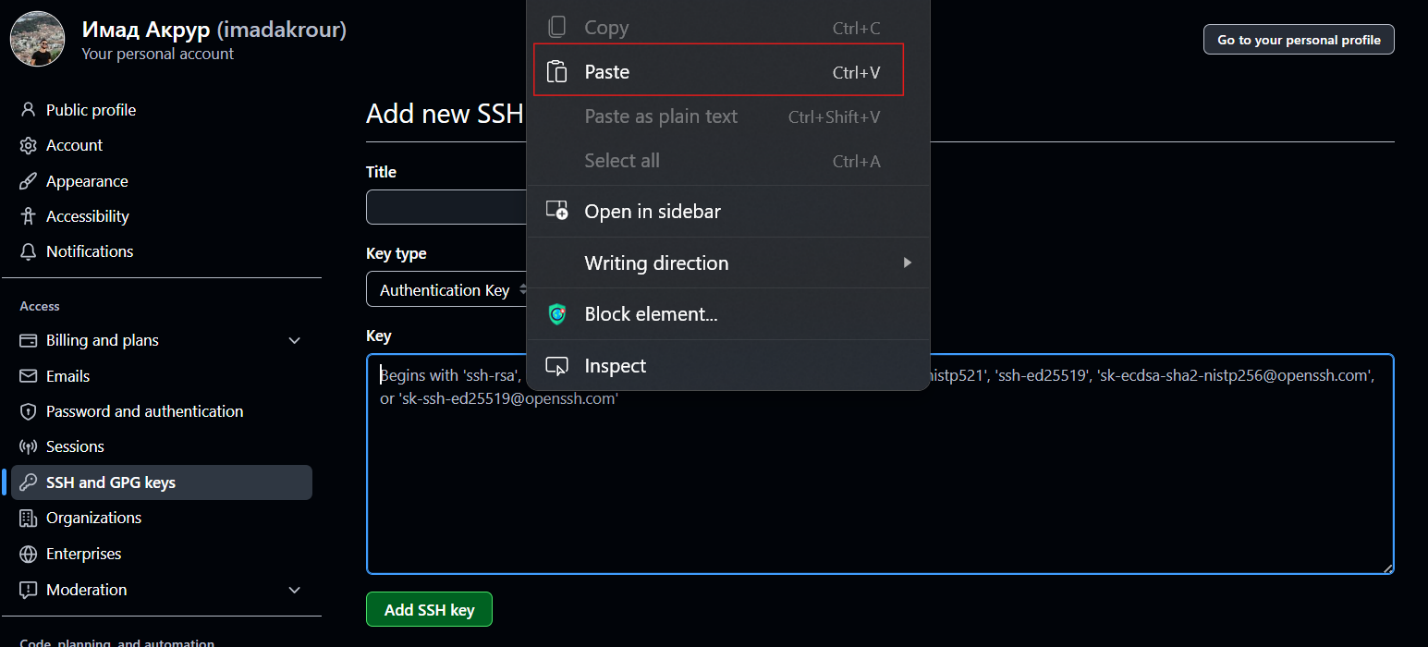


Рис. 6: рисунок 6

*вставить ключ на сайт GitHub.*

#### 4.2.2.4 Создание рабочего пространства и репозитория курса на основе шаблона

Для правильной организации рабочих файлов и проектов в рамках курса необходимо создать рабочее пространство по определенной структуре. В этом шаге будет выполнено создание директории для предмета «Архитектура компьютера», а также будет продемонстрировано, как структурировать папки для лабораторных работ.

**Описание выполняемого задания:**

**Создание структуры рабочего пространства :**

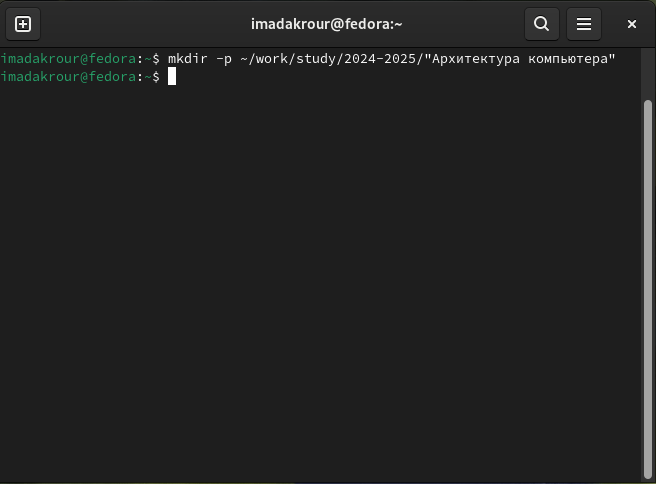


Рис. 7: рисунок 7

*Эта команда создаст нужные каталоги по иерархии. Опция -p создает промежуточные каталоги, если они еще не существуют*

#### 4.2.2.5 Сознание репозитория курса на основе шаблона



Рис. 8: рисунок 8

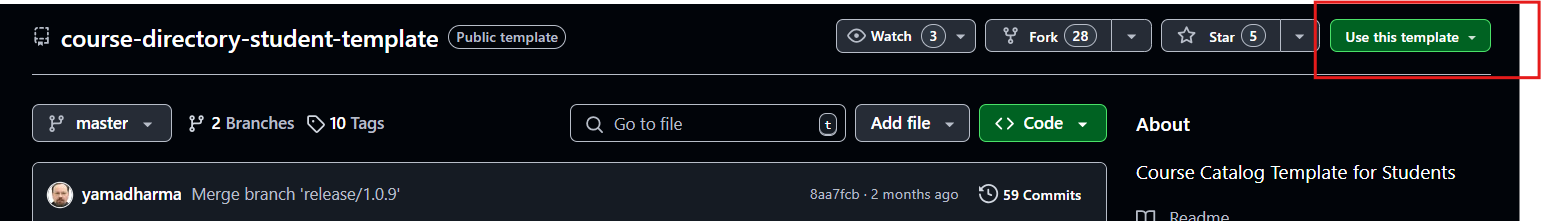


Рис. 9: рисунок 9

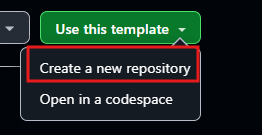


Рис. 10: рисунок 10

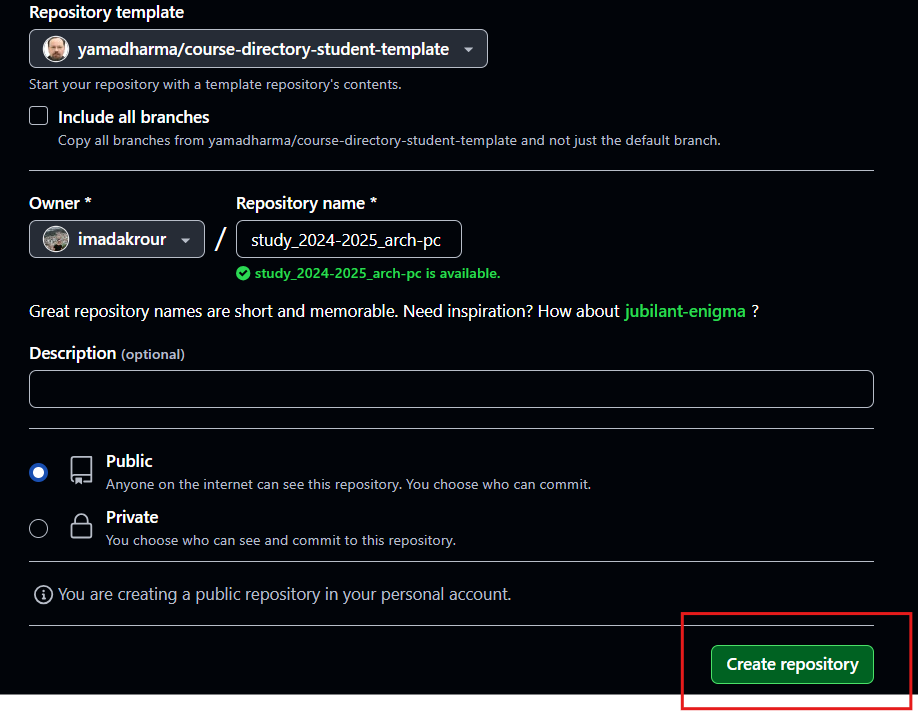


Рис. 11: рисунок 11

*Эти действия создают новый репозиторий на основе предоставленного шаблона, который содержит структуру и необходимые файлы для работы по курсу.*

**Клонирование репозитория на локальный компьютер :**

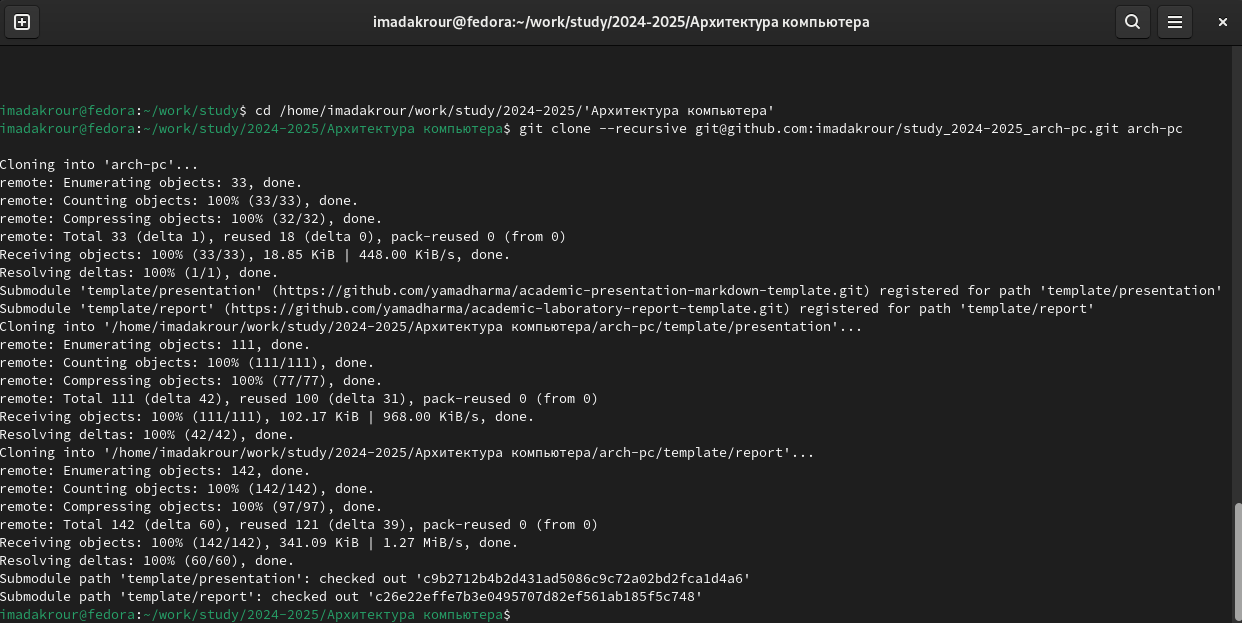


Рис. 12: рисунок 12

*Команда git clone –recursive позволяет загрузить все файлы из удаленного репозитория в папку arch-pc.*

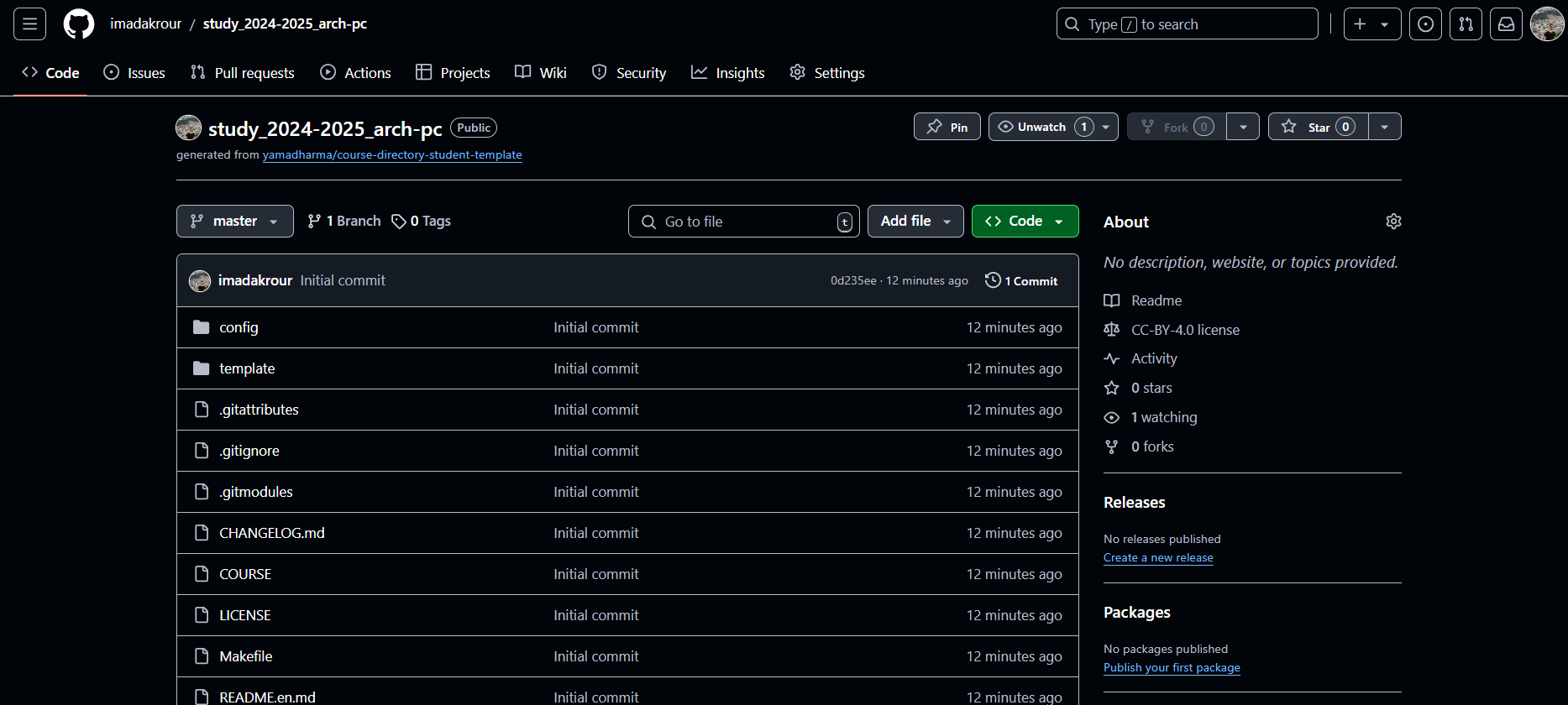


Рис. 13: рисунок 13

**Настройка каталога курса :**

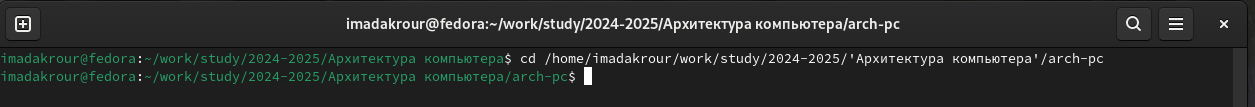


Рис. 14: рисунок 14

**Удаление ненужного файла package.json** **:**

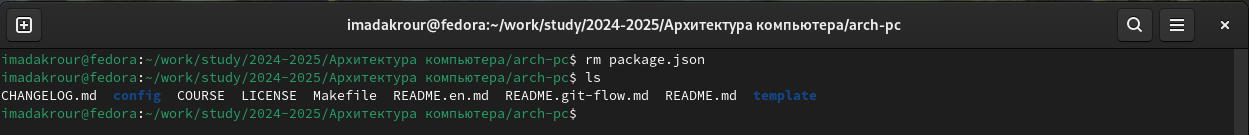


Рис. 15: рисунок 15

*Удаление лишних файлов помогает избежать путаницы и оставить только необходимые для курса файлы.*

**Создание файла COURSE с названием курса:**

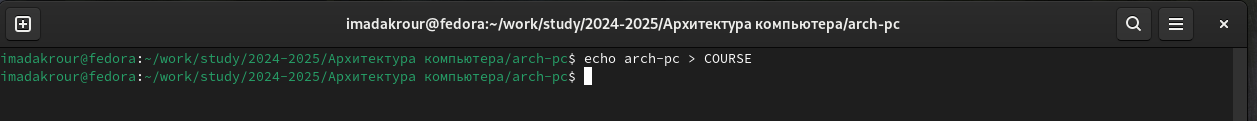


Рис. 16: рисунок 16

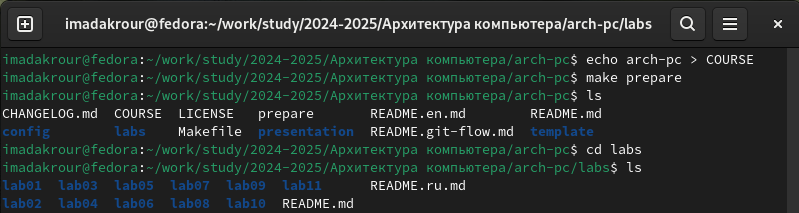


Рис. 17: рисунок 17

**Отправка изменений на сервер :**

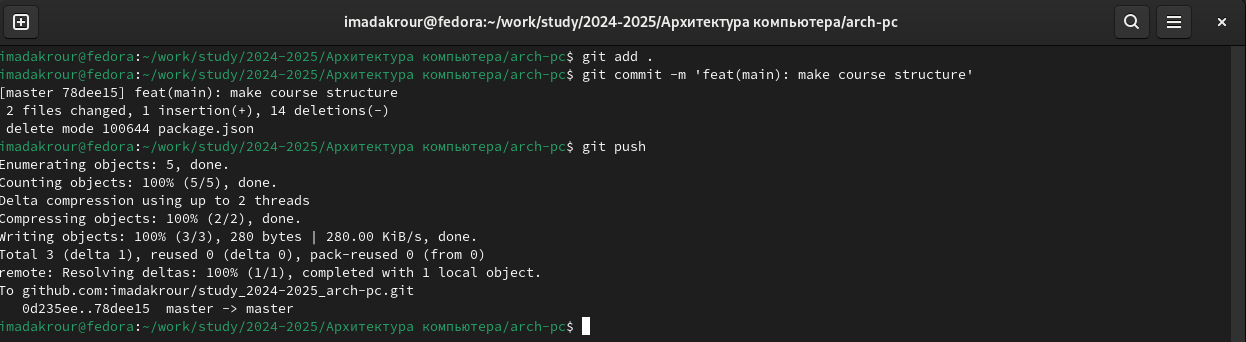


Рис. 18: рисунок 18

*Эти команды добавляют изменения в локальный репозиторий, создают коммит с описанием и отправляют изменения в удаленный репозиторий на GitHub.*

## 4.3 1. Написание отчёта в формате Markdown

Сначала я создал файл под названием report.md в текстовом редакторе. Этот файл содержит весь материал для лабораторной работы №2, оформленный в формате Markdown

|  |
| --- |
| Рис. 19: рисунок 23 |

Рис. 19: рисунок 23

## 4.4 2. Генерация форматов PDF, DOCX и MD

Когда файл Markdown был готов, я с помощью команды make report сгенерировал отчёт в трёх форматах: Markdown (.md), PDF (.pdf) и Word (.docx).

|  |
| --- |
| Рис. 20: рисунок 24 |

Рис. 20: рисунок 24

## 4.5 3. Отправка файлов в репозиторий

После создания файлов я зафиксировал изменения и отправил их в свой Git-репозиторий.

|  |
| --- |
| Рис. 21: рисунок 25 |

Рис. 21: рисунок 25

## 4.6 Описание результатов выполнения заданий для самостоятельной работы

### 4.6.1 Создание отчета в соответствующем каталоге рабочего пространства :

**Описание задания:**

Данное задание включает в себя создание отчета о выполнении лабораторной работы, копирование предыдущих отчетов и загрузку файлов на GitHub.

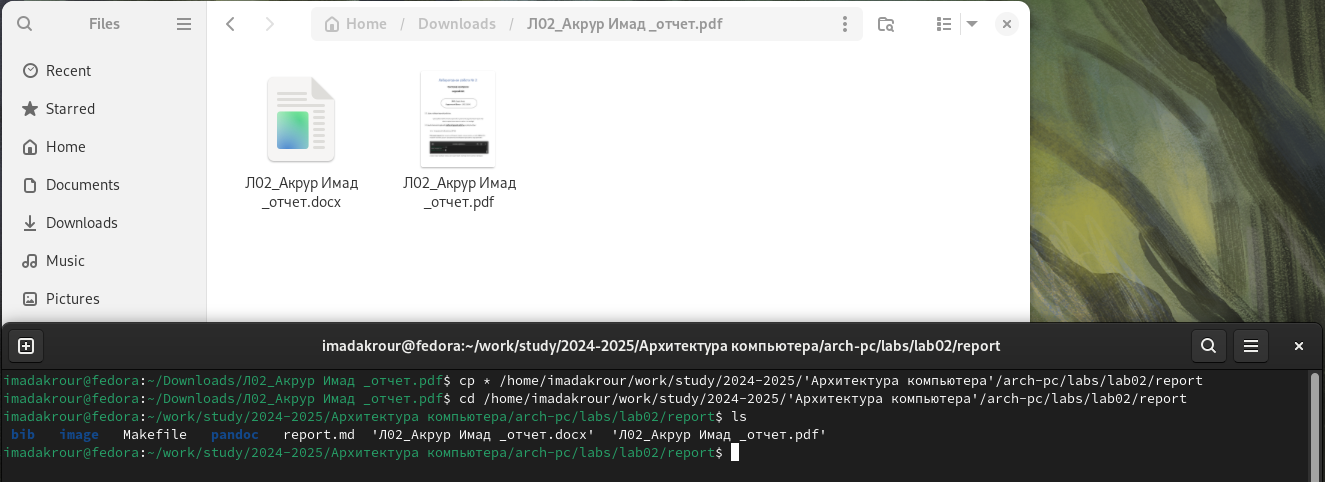


Рис. 22: рисунок 19

*Я скопировал файл отчета из lab01 в lab01/report, а файл отчета из lab02 в lab02/report, используя команду cp. Сначала я перешел в каталог с файлами отчета с помощью команды cd, а затем выполнил команду для копирования всех файлов*

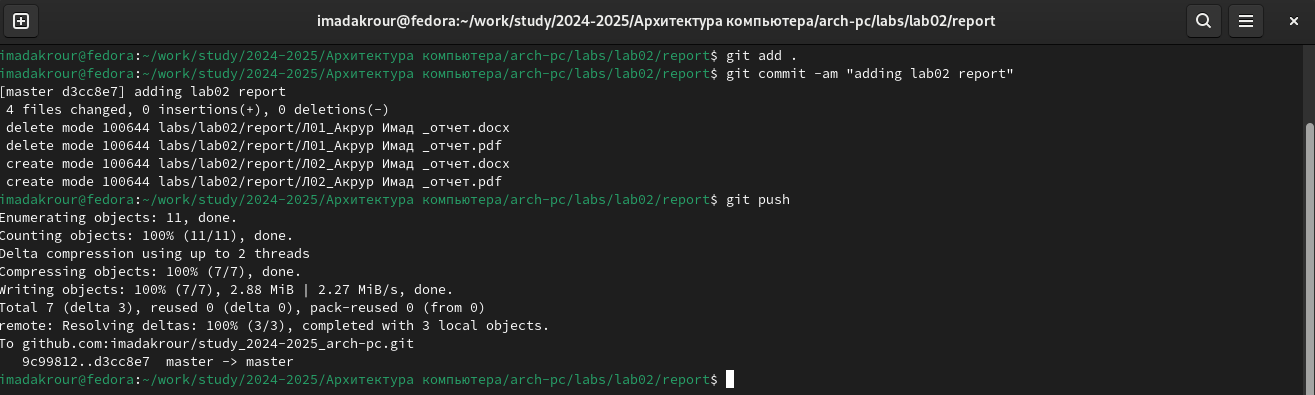


Рис. 23: рисунок 20

*После копирования файлов я выполнил команды для отправки их на* ***GitHub***



Рис. 24: рисунок 21

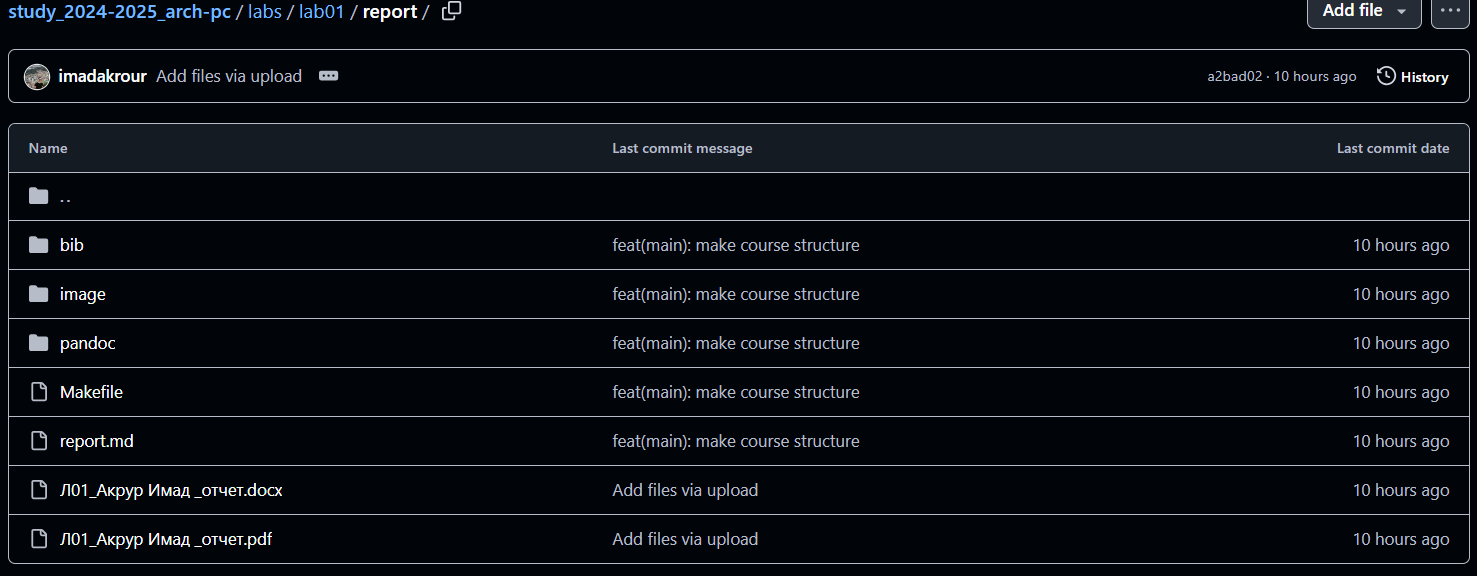


Рис. 25: рисунок 22

*Затем я сделал скриншот в GitHub, чтобы показать, что файлы были обновлены и теперь отображаются в репозито*

## 4.7 выводы по результатам выполнения заданий: :

Выполнение задания помогло закрепить навыки работы с системами контроля версий, организацией файлов в рабочем пространстве, а также загрузкой проектов на GitHub. Эти навыки важны для поддержания эффективной командной работы, обеспечения безопасности данных и удобства доступа к прошлым версиям проектов.

# 5 Выводы

* к концу лабораторной работы этой лабораторной работе мы узнали, как использовать markdown для создания pdf-файлов быстрее и эффективнее.