Лабораторная работа №2

Первоначальная настройка GitHub

Губайдуллина Софья Романовна

Содержание

# 1 Цель работы

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

# 2 Задание

1. Базовая настройка git и примеры использования;
2. Создание ключа SSH;
3. Верификация коммитов с помощью PGP;
4. Настройка автоматических подписей коммитов git;
5. Настройка gh;
6. Настройка каталога курса;
7. Ответы на контрольные вопросы.

# 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. 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. 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. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями.

## 3.1 Коммиты

Коммиты имеют следующие свойства: - author (автор) — контрибьютор, выполнивший работу (указывается для справки); - committer (коммитер) — пользователь, который закоммитил изменения. Для подписывания коммитов используется технология PGP. Подпись коммита позволяет удостовериться в том, кто является коммитером. Авторство не проверяется.

# 4 Выполнение лабораторной работы

1. Начинаю лабораторную работу №3, подразумевающую в себя отчет по выполнению работы №2 с настройки git.
2. Так как свой аккаунт в GitHub у меня уже был, мне было необходимо лишь создать новый SSH-ключ в терминале и копировать его в code своего аккаунта в GitHub (рис. 1).

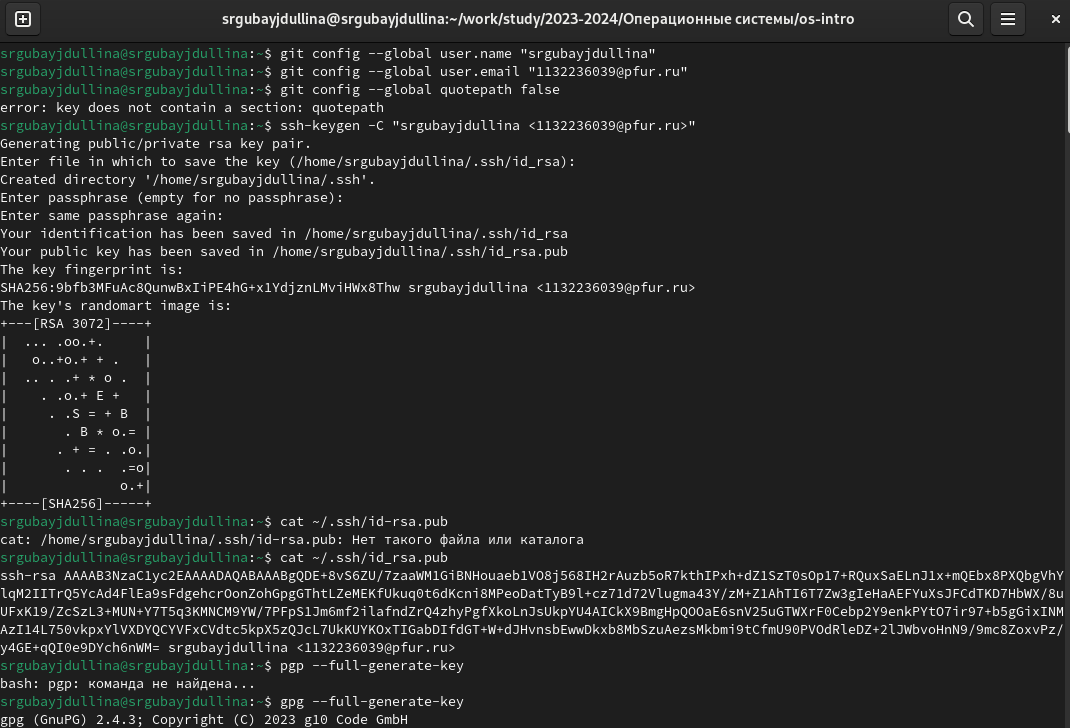


Рис. 1: Настройка git и получение нового ключа SSH

1. После по ходу лабораторной работы мне потребовалось создать PGP-ключ, и так же копировать его в аккаунт GitHub (рис. 2). Для этого я вывела список ключей и скопировала отпечаток приватного ключа.

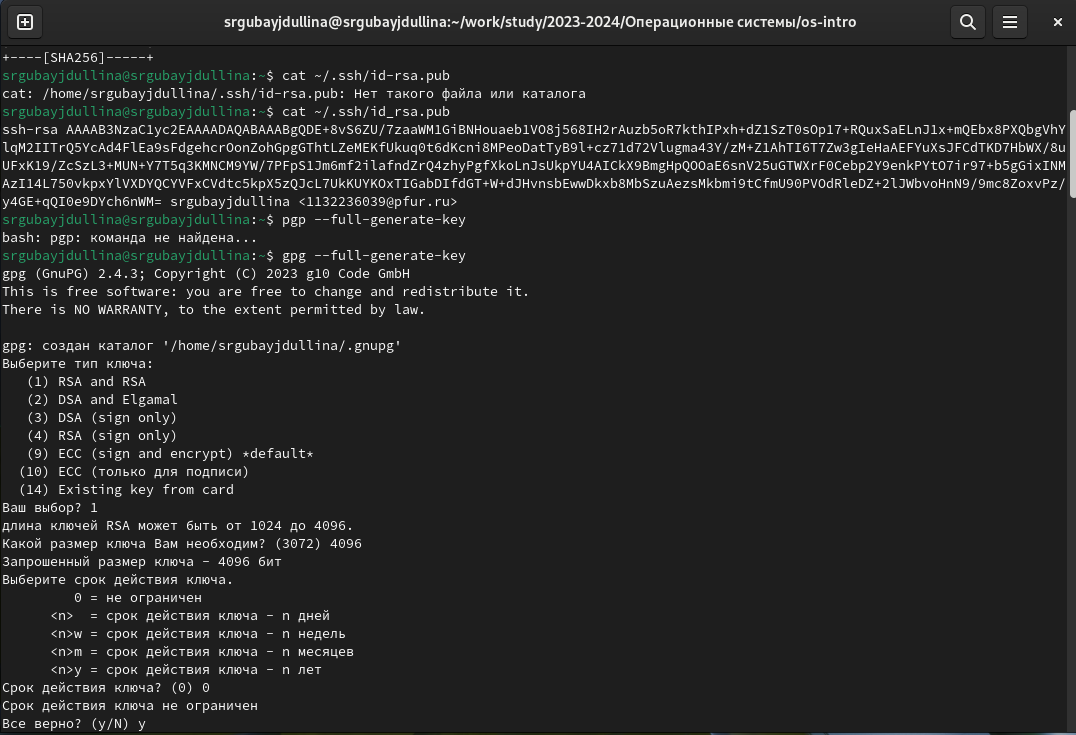


Рис. 2: Создание ключа PGP

Команды в терминале привели к следующему выводу (идентифицирование ключа) (рис. 3) (рис. 4) (рис. 5).

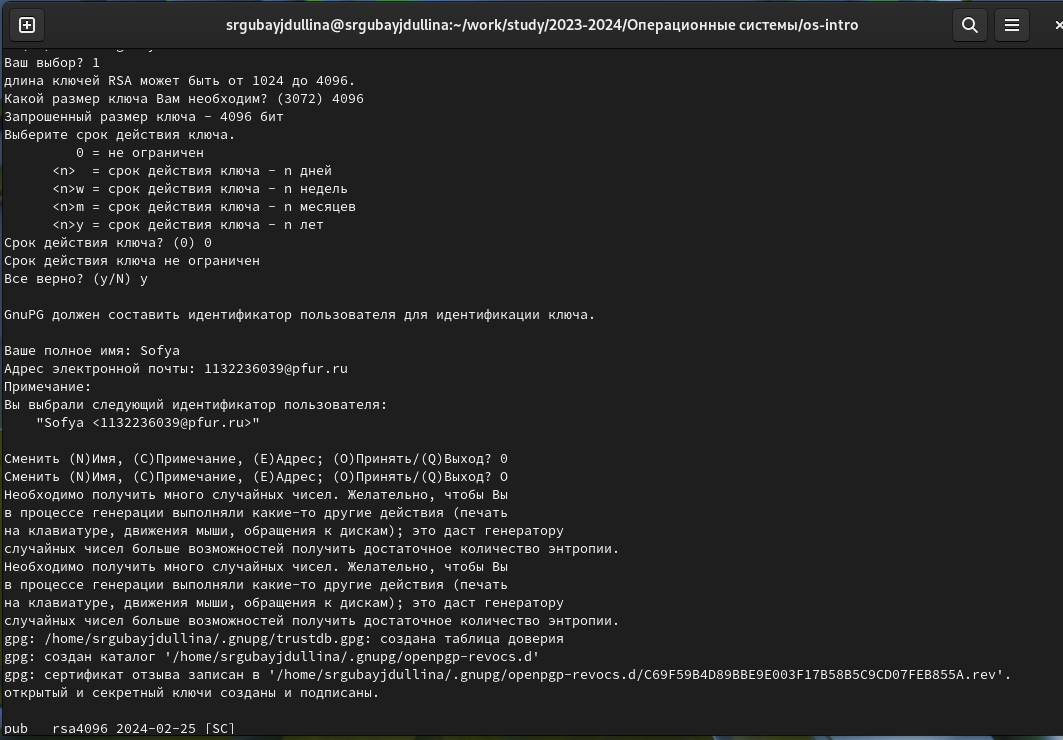


Рис. 3: Идентификация ключа PGP

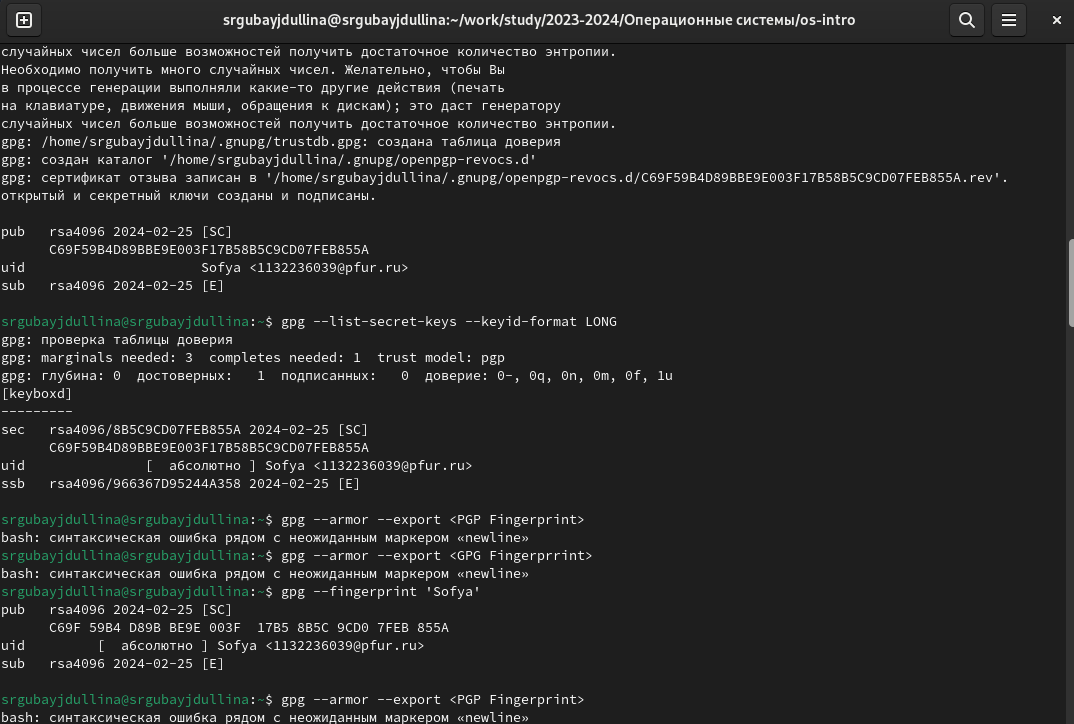


Рис. 4: Вывод данных PGP-ключа

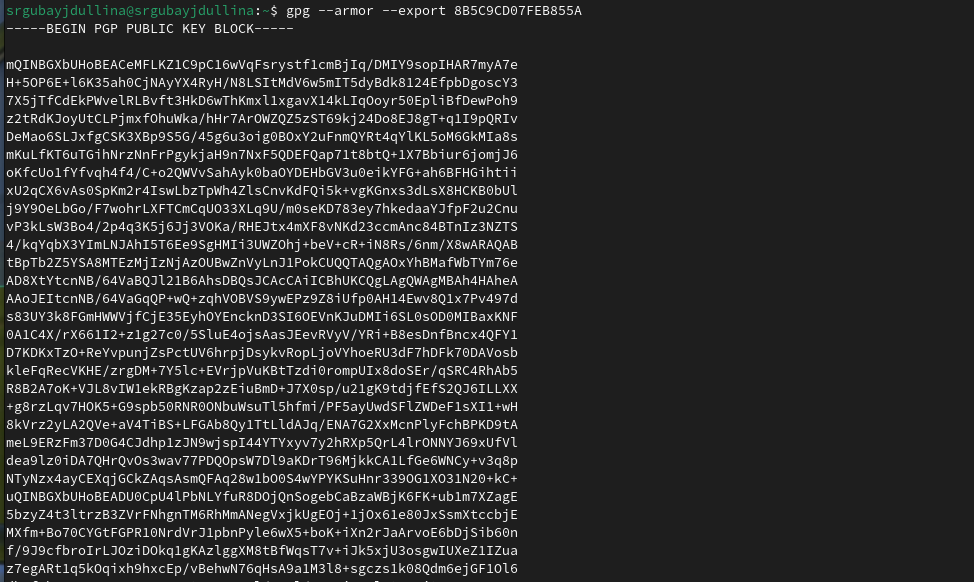


Рис. 5: Отпечаток PGP-ключа

1. После мне нужно было настроить автоматические подписи коммитов, используя свой email, а так же настроить gh.
2. Для этого авторизуюсь и ввожу указанный на экране код в окно GitHub браузера (рис. 6).

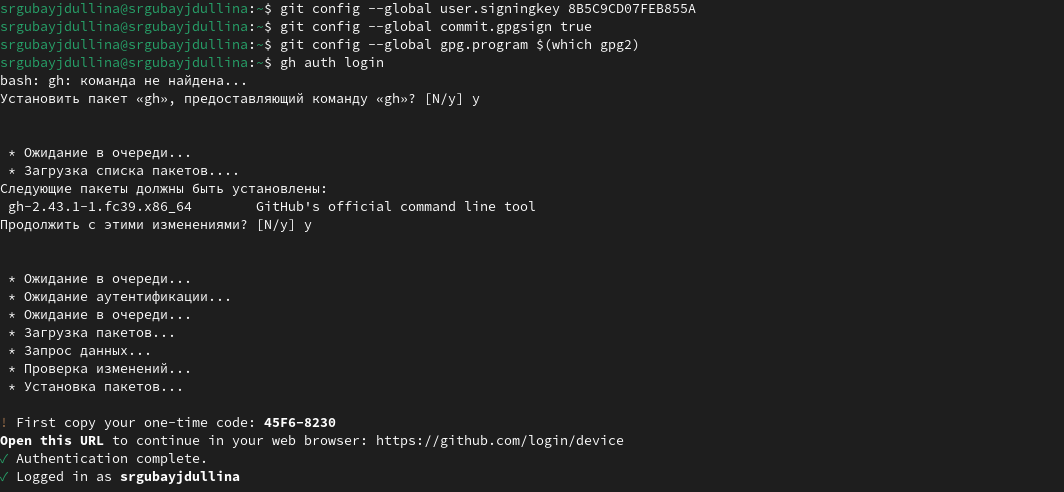


Рис. 6: Авторизация в Github и добавление репозитория

1. Далее начинаю создавать шаблон рабочего пространства. Создаю католог и соответствующие папки “work”, “study”, “Операционные системы” и т.д. (рис. 7). Все вышесозданное тут же клонирую на GitHub (рис. 8).

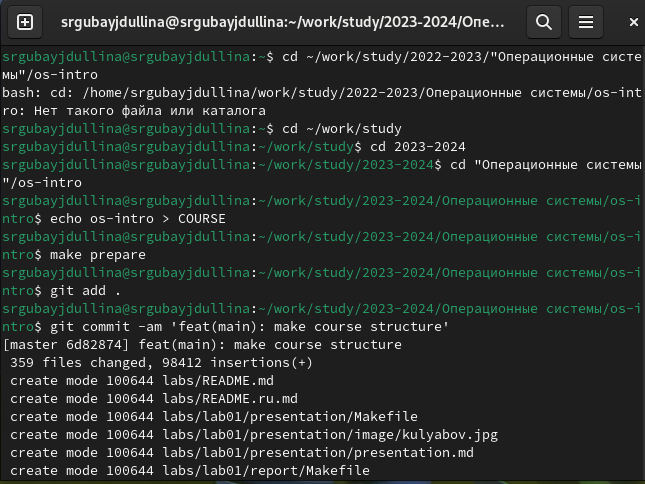


Рис. 7: Создание и добавление нового репозитория

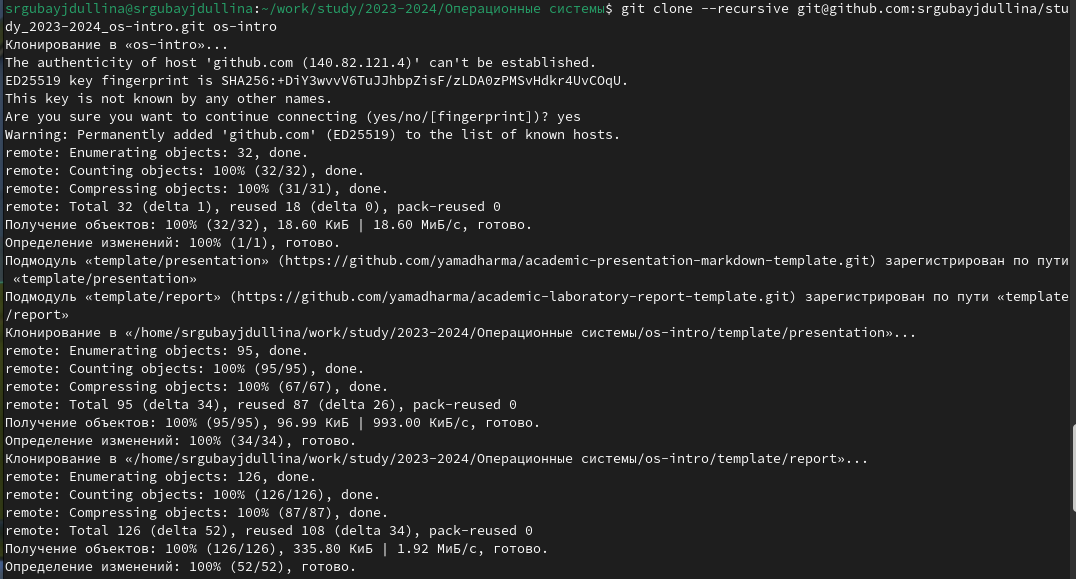


Рис. 8: Создание нового репозитория

Перехожу к настройке самого каталога курса. Перехожу в него, удаляю лишние файлы, создаю необходимые каталоги и отправляю файлы на сервер (рис. 9) (рис. 10)

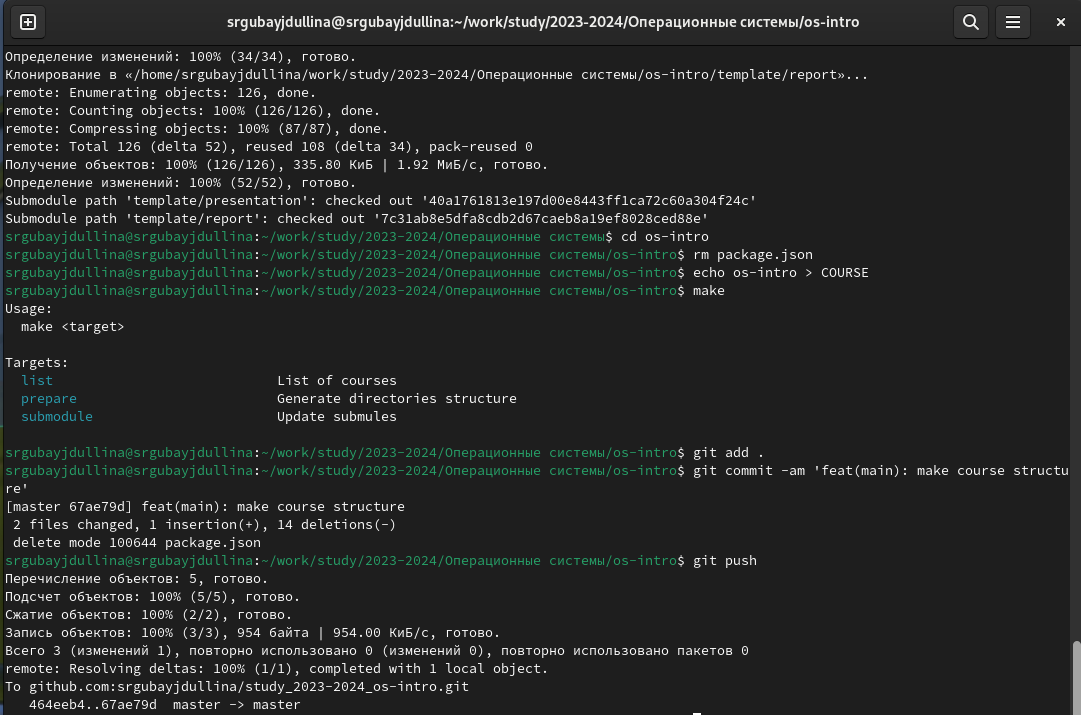


Рис. 9: Создание курса

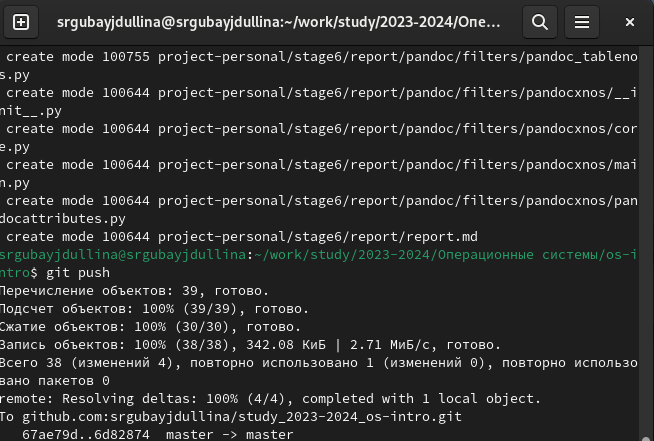


Рис. 10: Создание курса

Проверяю как был создан новый репозиторий и новые курсы (рис. 11) (рис.12)

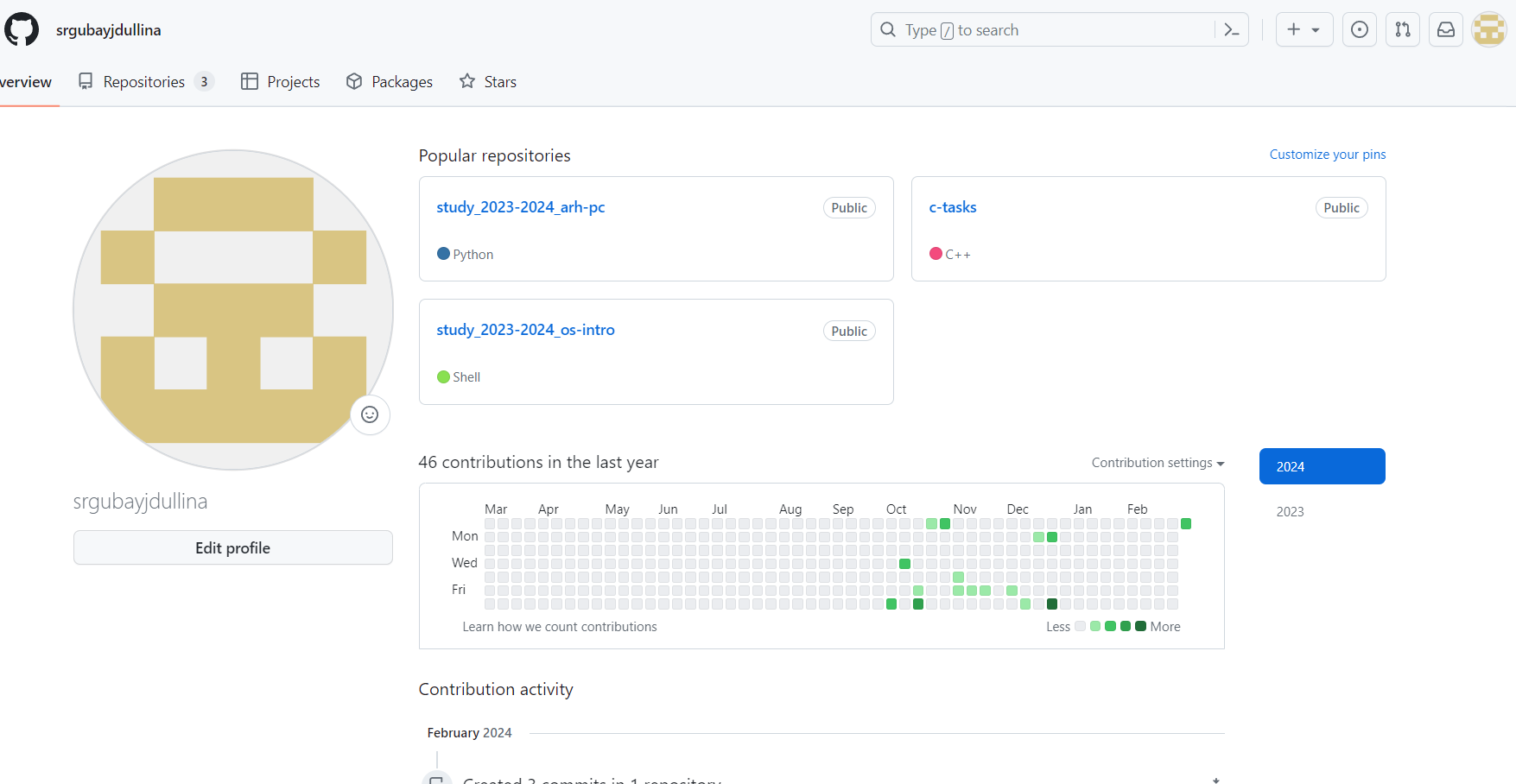


Рис. 11: Проверка созданного репозитория

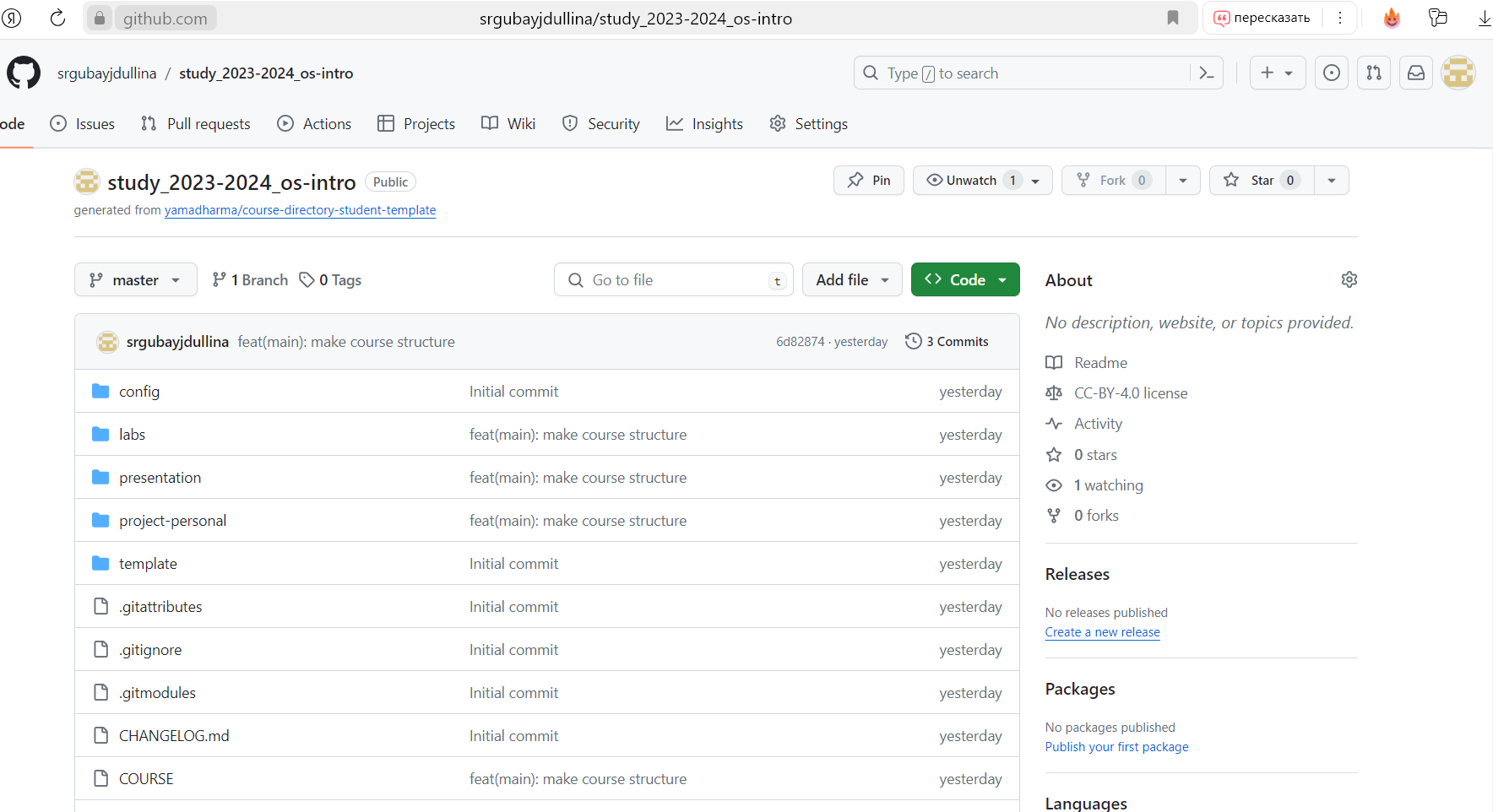


Рис. 12: Проверка созданных курсов

1. Ответы на вопросы.
2. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Это специальное программное обеспечение для облегчения работы с изменяющейся информацией. 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.
3. Хранилище - основное место (сервер) для хранения всех изменений файлов (файлов и их изменений). Сommit - сохранение всех добавленных изменений в файлах. История - все изменения файлов, Рабочая копия - определенный снимок версии файла.
4. Централизованные системы контроля версий предполагают наличие единого репозитория для хранения файлов, в то время как децентрализованные могут иметь несколько репозиториев. Среди классических (централизованных) VCS наиболее известны CVS, Subversion, а среди распределённых (децентрализованных) — Git, Bazaar, Mercurial.
5. 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.
6. Обновление копий, модификация файлов, сохранение (коммит) всех изменений, отправка изменений на сервер.
7. 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.
8. Работа с ветками - git branch, git checkout; сохранение изменений и комментариев - git commit; отправка на сервер - git push; обновление - git pull; добавление файлов - git add.
9. Локальный репозиторий - репозиторий, находящийся в нашем ПК для быстрого доступа к нему. Удаленный же репозиторий - репозиторий, находящийся на другом сервере, чтобы получить возможность удаленно работать с ним при неимении возможности внести изменения через локальный репозиторий.
10. Ветки необходимы для того, чтобы сразу несколько пользователей могли вести работу над одим и тем же проектом.
11. Игнорирования в git необходимы для того, чтобы нежелательные (ненужные) файлы не попали на сервер при коммитах.

# 5 Выводы

Выполняя лабораторную работу по настройке git, я подробно изучила идеологию и применение средств контроля версий и освоила новые умения по работе с git.