Лабораторная работа №2

Дисциплина: Операционные системы

Ищенко Ирина Олеговна

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

Установили git и gh (рис. [1](#fig:001)).

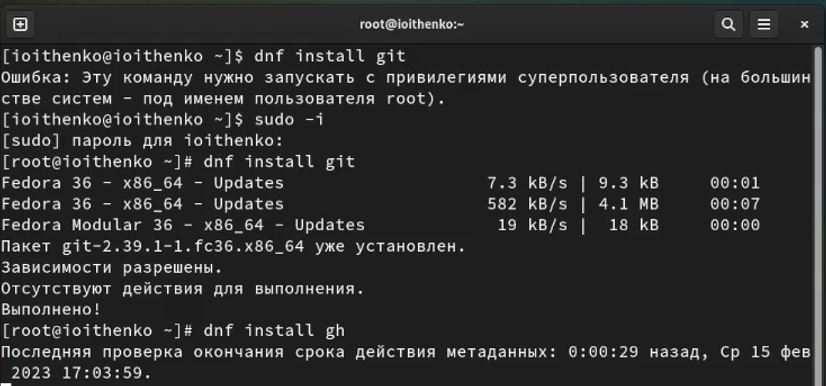


Figure 1: Установка git и gh

Зададим имя и email владельца репозитория. Настроим utf-8 в выводе сообщений git. Зададим имя начальной ветки. Задаем параметры autocrlf и safecrlf (рис. [2](#fig:002)).

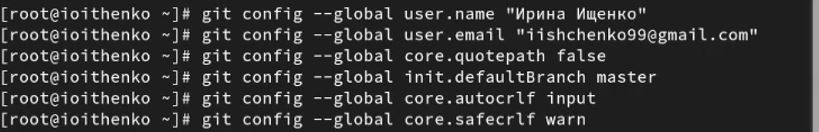


Figure 2: Настройка параметров

Создаем ключи ssh: - по алгоритму rsa с ключём размером 4096 бит (рис. [3](#fig:003)).

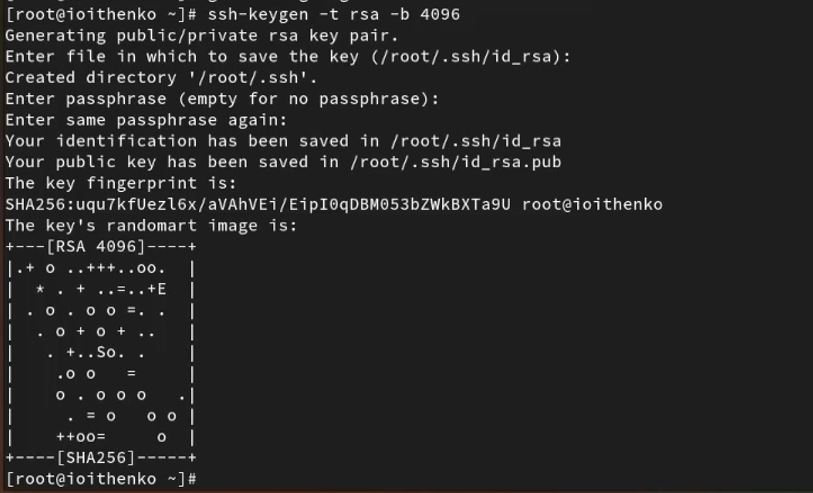


Figure 3: ssh ключ

* по алгоритму ed25519 (рис. [4](#fig:004)).

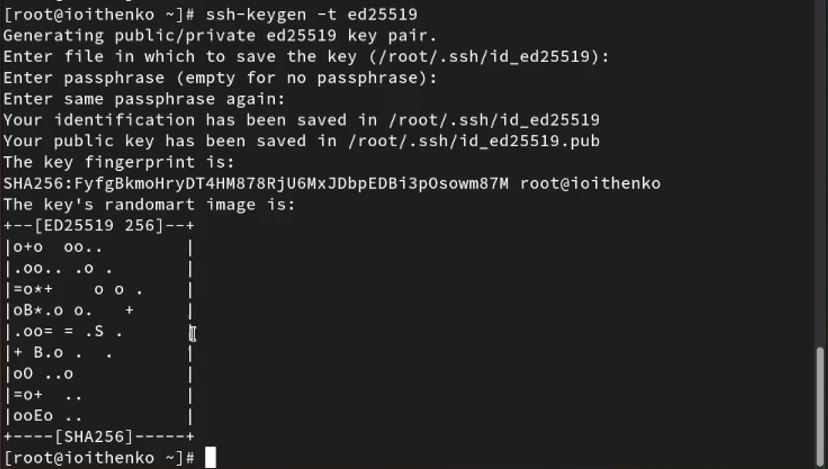


Figure 4: ssh ключ

Генерируем ключ pgp: тип RSA and RSA; размер 4096; срок действия - 0 (рис. [5](#fig:005)). Указываем имя и адрес электронной почты (рис. ??).

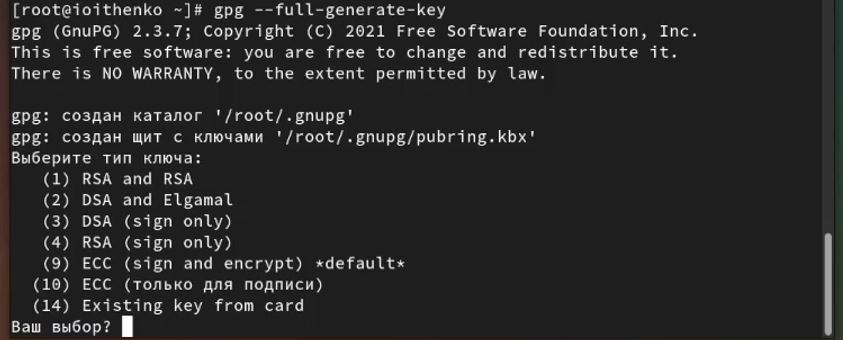


Figure 5: pgp ключ

Учетная запись на github настроена с первого семестра.

Выводим список ключей и копируем отпечаток приватного ключа. Cкопируем наш сгенерированный PGP ключ в буфер обмена (рис. [6](#fig:007)).

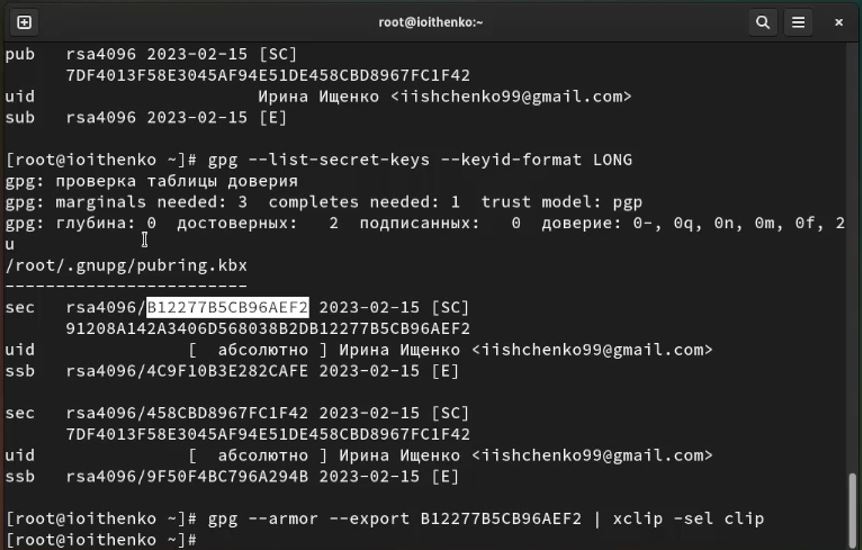


Figure 6: Копирование ключа

В настройках github добавляем полученный ключ (рис. [7](#fig:008)).

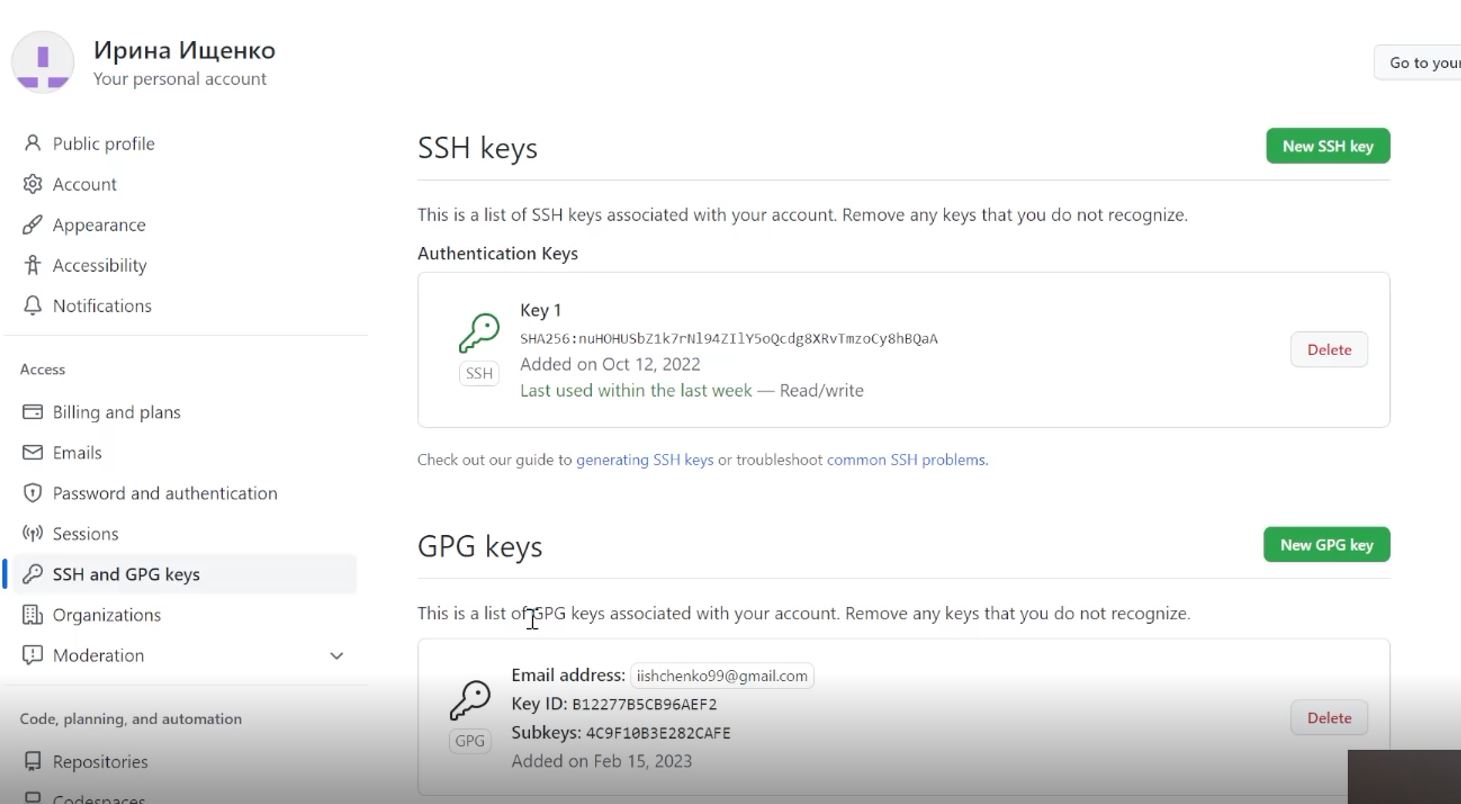


Figure 7: Добавление ключа

Используя введёный email, укажем Git применять его при подписи коммитов (рис. [8](#fig:009)).

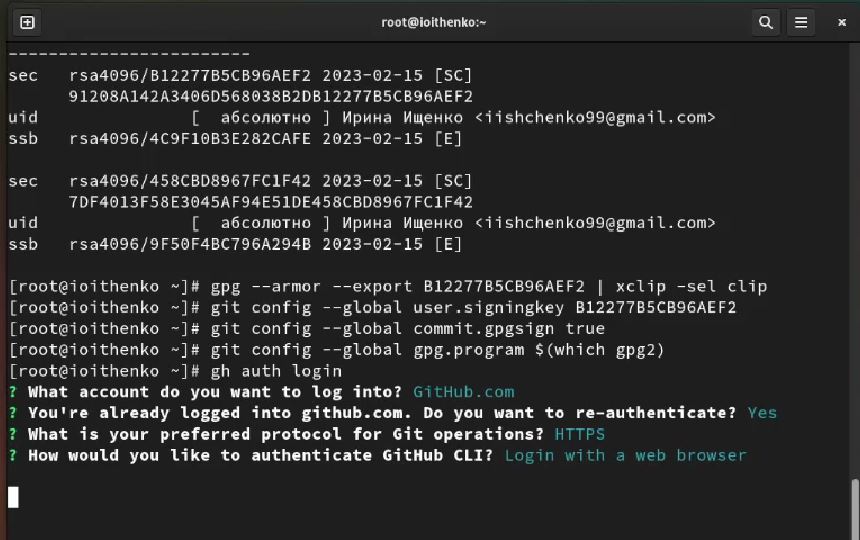


Figure 8: Настройка подписей коммитов

Авторизовываемся (рис. [9](#fig:0010)).

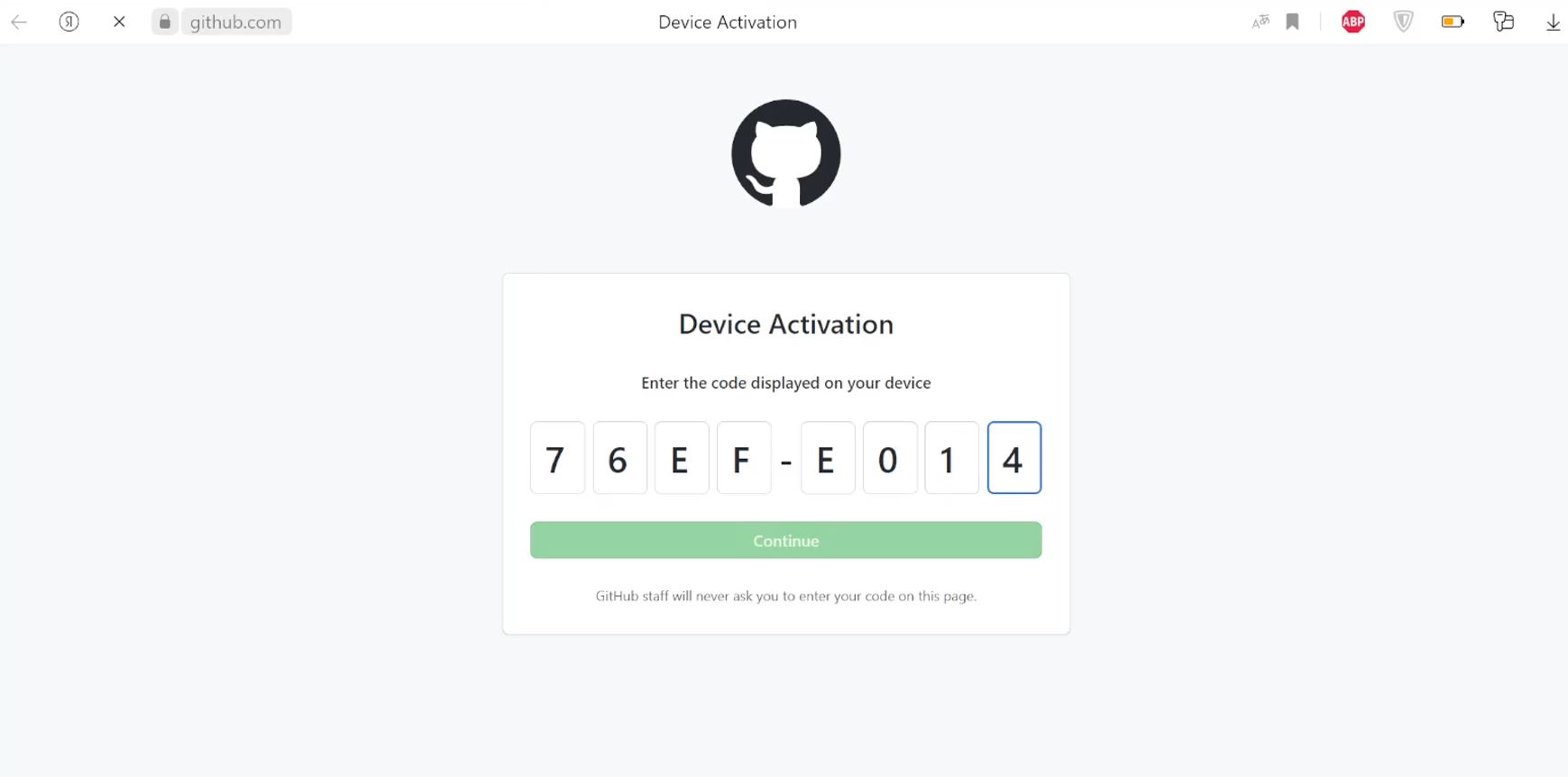


Figure 9: Авторизация

С помощью шаблона создаем новый репозиторий (рис. [10](#fig:0011)).

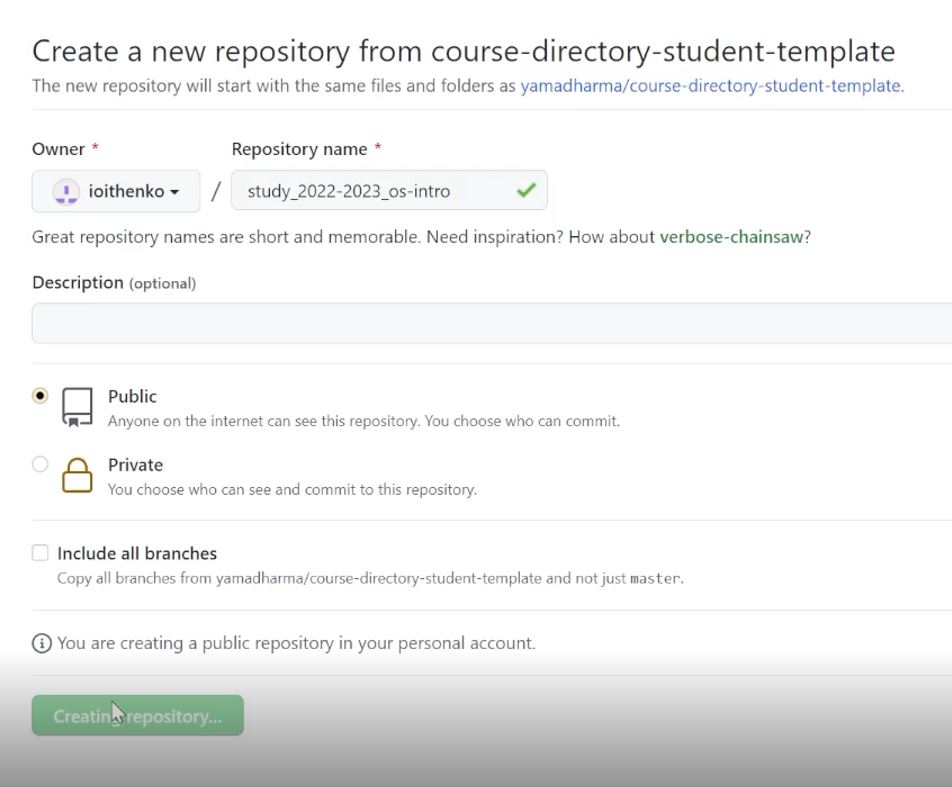


Figure 10: Создание репозитория

Создаем каталог “Операционные системы”, переходим в него, создаем структуру с помощью шаблона, клонируем репозиторий (рис. [11](#fig:0012)).

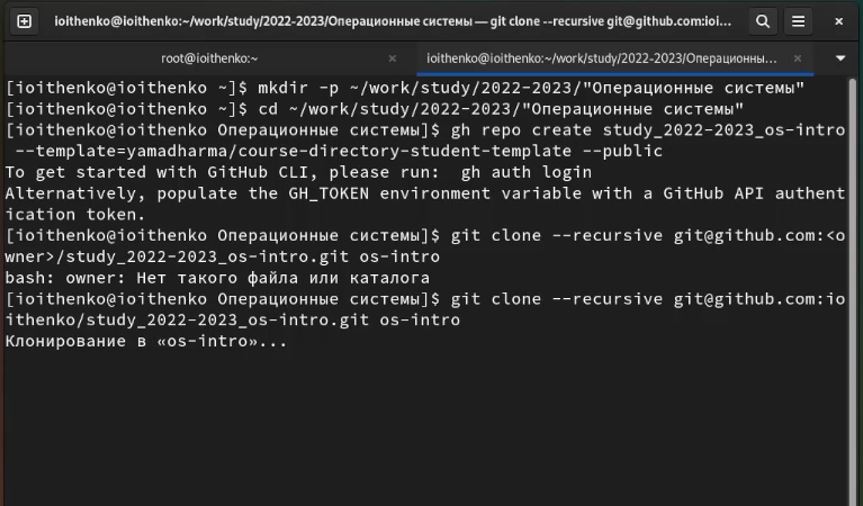


Figure 11: Создание репозитория

Удаляем лишний каталог. Создаем необходимые каталоги (рис. [12](#fig:0013)).

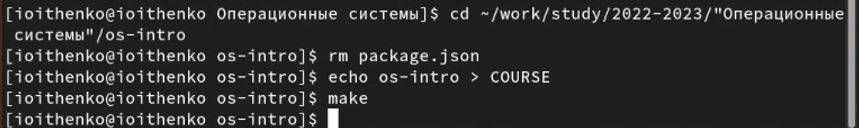


Figure 12: Удаление и добавление каталогов

Отправляем файлы на сервер (рис. [13](#fig:0014)) и (рис. [14](#fig:0015)).

Figure 13: Отправление файлов на сервер

Figure 13: Отправление файлов на сервер

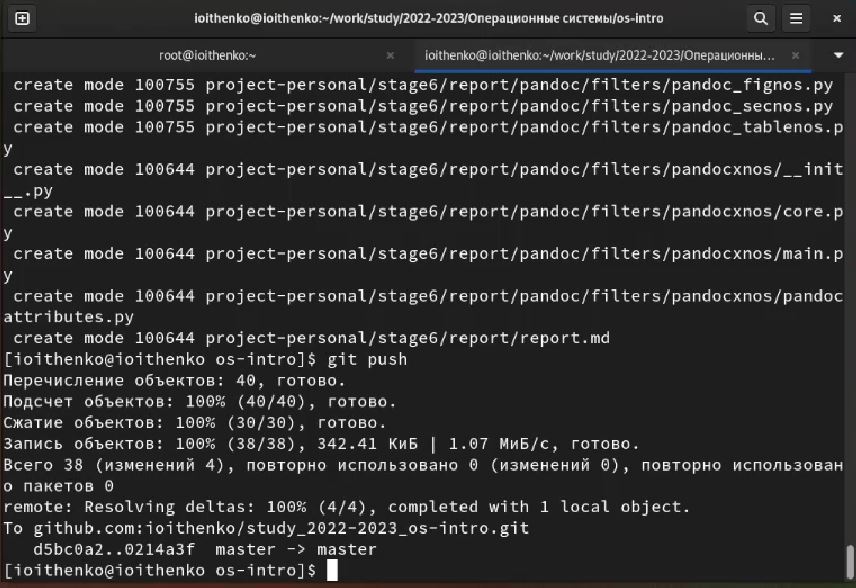


Figure 14: Отправление файлов на сервер

Репозиторий настроен (рис. [15](#fig:0016)).

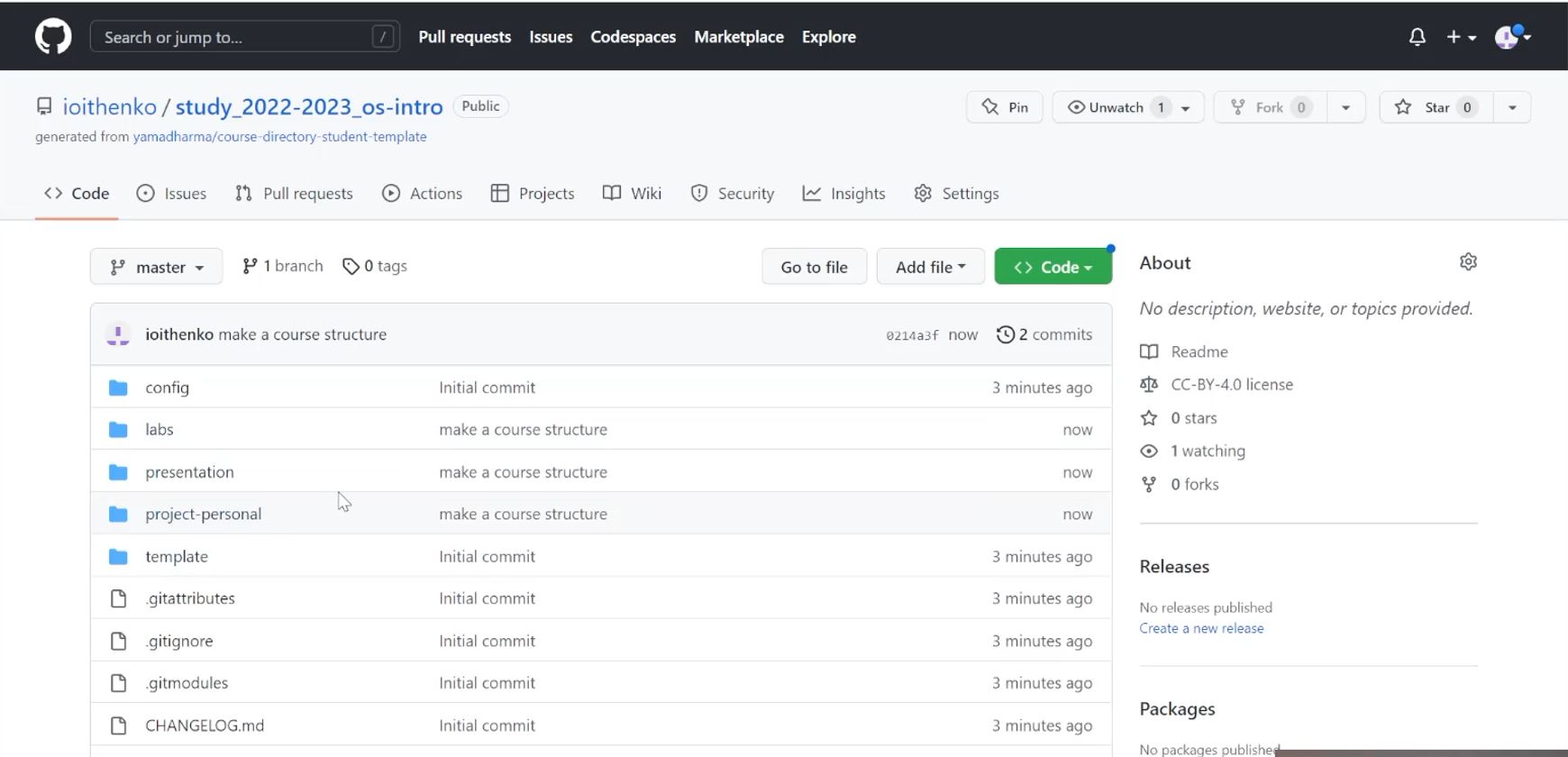


Figure 15: Итоговый вид репозитория

# 3 Ответы на контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Система контроля версий — это программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени. VCS прмменяется при работе н6ескольких разработчиков над одним общим проектом. Система контроля версий хранит все изменения, внесенные в проект, а также разработчики могут вернуться к любой более ранней версии проекта.

1. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Хранилище - место расположения файлов и каталогов проекта, изменения которых отслеживаются. Коммит - операция отправки в репозиторий измениний, которые внес разработчик в свою копию проекта. История - информация о всех более ранних версиях проекта. Рабочая копия - текущее состояние файлов проекта, полученных из хранилища и используемых разработчиков в настоящее время.

1. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованная система контроля версий предназначена для решения основной проблемы локальной системы контроля версий. Для организации такой системы контроля версий используется единственный сервер, который содержит все версии файлов. Пример централизованной VCS - CVS, Subversion, Perforce. Децентрализованные системы контроля версий - это такие системы, как Git, Mercurial, Bazaar или, например, Darcs. Каждый пользователь имеет свою версию репозитория, может вносить в нее изменения.

Различие между централизованными и децентрализованными VCS заключается в количестве репозиториев.

1. Опишите действия с VCS при единоличной работе с хранилищем.

Разработчик работает с главной веткой master, при необходимости может создавать дополнительные ветки для отдельных задач проекта. При внесении изменений он сохраняет файлы в общем хранилище.

1. Опишите порядок работы с общим хранилищем VCS.

Каждый разработчик работает с отдельной веткой проекта и вносит изменения в локальный репозиторий. При завершении работы ему требуется запушить файлы на сервер, внести изменения в главную ветку.

1. Каковы основные задачи, решаемые инструментальным средством git?

Основные задачи: возможность коллективной работы над проектом и сохранение всей информации об изменениях, внесенных в проект.

1. Назовите и дайте краткую характеристику командам git.

Создание основного дерева репозитория: git init Получение обновлений (изменений) текущего дерева из центрального репозитория: git pull Отправка всех произведённых изменений локального дерева в центральный репозиторий: git push Просмотр списка изменённых файлов в текущей директории: git status Просмотр текущих изменений: git di добавить все изменённые и/или созданные файлы и/или каталоги: git add . добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add имена\_файлов удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена\_файлов сохранить все добавленные изменения и все изменённые файлы: git commit -am ‘Описание коммита’ сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit создание новой ветки, базирующейся на текущей: git checkout -b имя\_ветки переключение на некоторую ветку: git checkout имя\_ветки (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) отправка изменений конкретной ветки в центральный репозиторий: git push origin имя\_ветки слияние ветки с текущим деревом: git merge –no-ff имя\_ветки удаление локальной уже слитой с основным деревом ветки: git branch -d имя\_ветки принудительное удаление локальной ветки: git branch -D имя\_ветки удаление ветки с центрального репозитория: git push origin :имя\_ветки

1. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Локальный репозиторий используется одним человеком и расположен на его компьютере. Удаленный репозиторий может использоваться группой разработчиков, в центральный репозиторий они пушат из локальных репозиториев изменения, внесенные в проект.

1. Что такое и зачем могут быть нужны ветви (branches)?

Ветка - тэо указатель на один из коммитов. Ветки используются для работы над одной из частей проекта в отдельности от других его частей. Каждая ветка представляет собой отдельную копию проекта, что позволяет работать над разными версиями одного проекта.

1. Как и зачем можно игнорировать некоторые файлы при commit?

Игнорировать файлы при commit можно с помощью файла gitignor. Там помещаются файлы, не требующиеся в итоговой версии проекта.

# 4 Выводы

В ходе выполнения лабораторной работы я изучила идеологию и применение средств контроля версий и освоила умения по работе с git.

# Список литературы