Отчёт по лабораторной работе №2

Управление версиями

Грузинова Елизавета Константиновна

Содержание

Список иллюстраций

1 Цель работы

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

2 Задание

1. Создайте учётную запись на https://github.com.
2. Заполните основные данные на https://github.com.
3. Установка git-flow в Fedora Linux.
4. Установка gh в Fedora Linux.
5. Базовая настройка git.
6. Создайте ключи ssh.
7. Создайте ключи pgp.
8. Добавление PGP ключа в GitHub.
9. Настройка автоматических подписей коммитов git.
10. Настройка gh.
11. Создание репозитория курса на основе шаблона.
12. Настройка каталога курса.

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 дельтакомпрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

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. [1]

4 Выполнение лабораторной работы

1. Создайте учётную запись на https://github.com. (рис. 4.1)

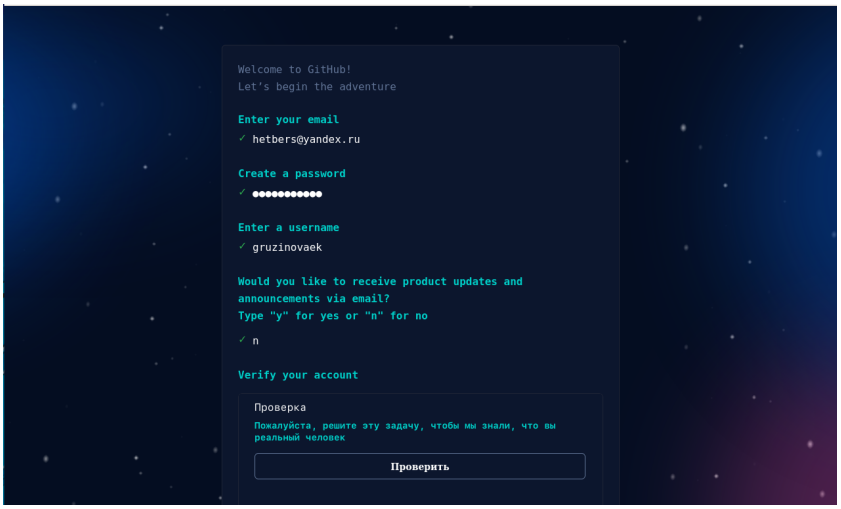

Рис. 4.1: Регистрация учетной записи на github.com

2. Заполните основные данные на https://github.com. (рис. 4.2)

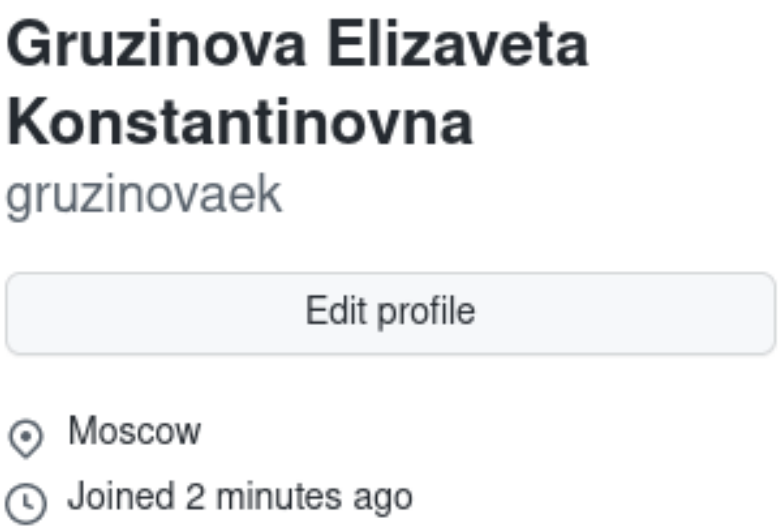

Рис. 4.2: Заполнение основных данных

3. Установка git-flow в Fedora Linux. (рис. 4.3, 4.4)

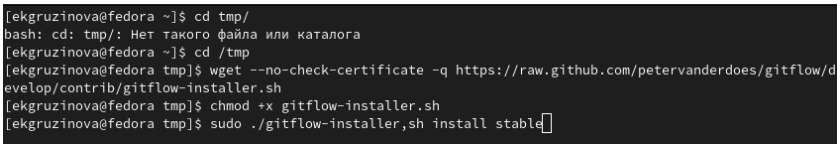

Рис. 4.3: Ручная установка git-flow

Рис. 4.4: Git-flow успешно установлен

4. Установка gh в Fedora Linux. (рис. 4.5)

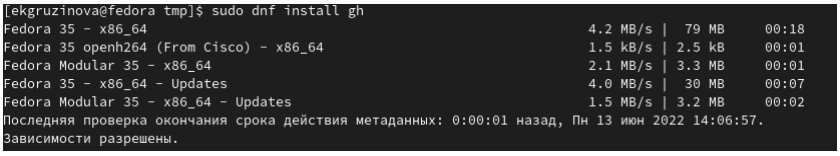

Рис. 4.5: Команда для установки gh в Fedora Linux

5. Базовая настройка git. (рис. 4.6, 4.7, 4.8, 4.9, 4.10)

```
[ekgruzinova@fedora tmp]$ git config --global user.name "Gruzinova Elizaveta Konstantinovna"
[ekgruzinova@fedora tmp]$ git config --global user.email "hetbers@yandex.ru"
```

Рис. 4.6: Установка имени и email владельца репозитория

```
[ekgruzinova@fedora tmp]$ git config --global core.quotepath false
```

Рис. 4.7: Настройка utf-8 в выводе сообщений git

```
[ekgruzinova@fedora tmp]$ git config --global init.defaultBranch master
```

Рис. 4.8: Установка имени начальной ветки master

```
[ekgruzinova@fedora tmp]$ git config --global core.autocrlf input
```

Рис. 4.9: Регулирование параметра autocrlf

```
[ekgruzinova@fedora tmp]$ git config --global core.safecrlf warn
```

Рис. 4.10: Регулирование параметра safecrlf

6. Создайте ключи ssh. (рис. 4.11, 4.12)

```
[ekgruzinova@fedora tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ekgruzinova/.ssh/id_rsa): ksfec12rt
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ksfec12rt
Your public key has been saved in ksfec12rt.pub
The key fingerprint is:
SHA256:HnvVTdzzKLNDVpq2YGuu2d/F5gf1GGdzoPYRIQUyG5U ekgruzinova@fedora
The key's randomart image is:
+---[RSA 4096]----+
|          +.++o. |
|           =E.o..|
|          .  ..++|
|            o=o=B|
|        S o.X.oB*|
|       . + B =+..|
|        o + +  .+|
|         *   o +.|
|        o.o.. . o|
+----[SHA256]-----+
```

Рис. 4.11: Создание ключа ssh по алгоритму rsa с ключём размером 4096 бит

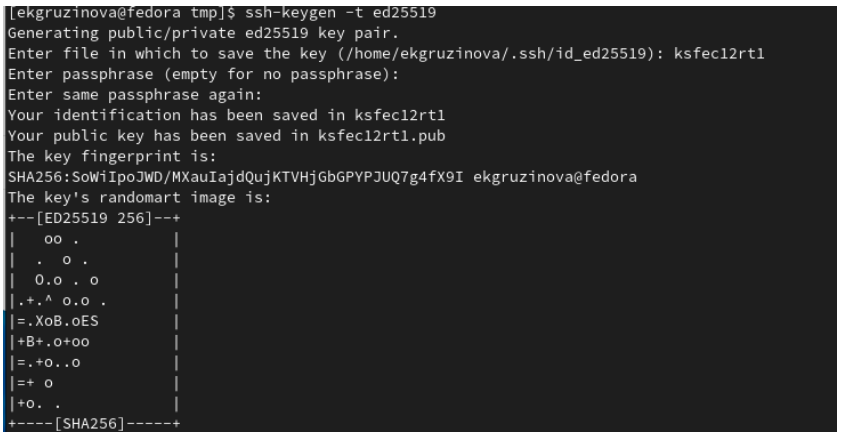

Рис. 4.12: Создание ключа ssh по алгоритму ed25519

7. Создайте ключи pgp. (рис. 4.13, 4.14, 4.15, 4.16)

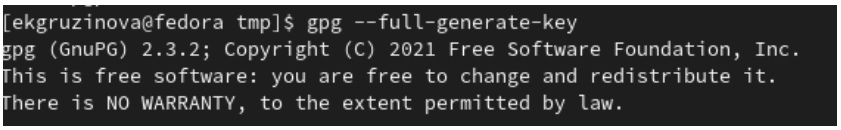

Рис. 4.13: Генерация ключа pgp

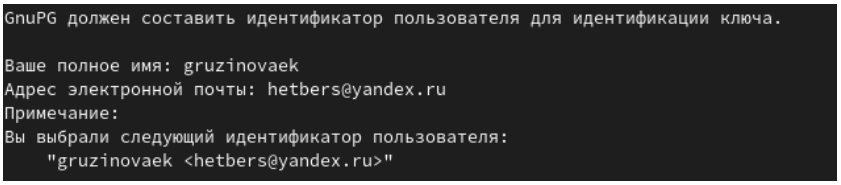

Рис. 4.14: Составление идентификатора пользователя для идентификации ключа

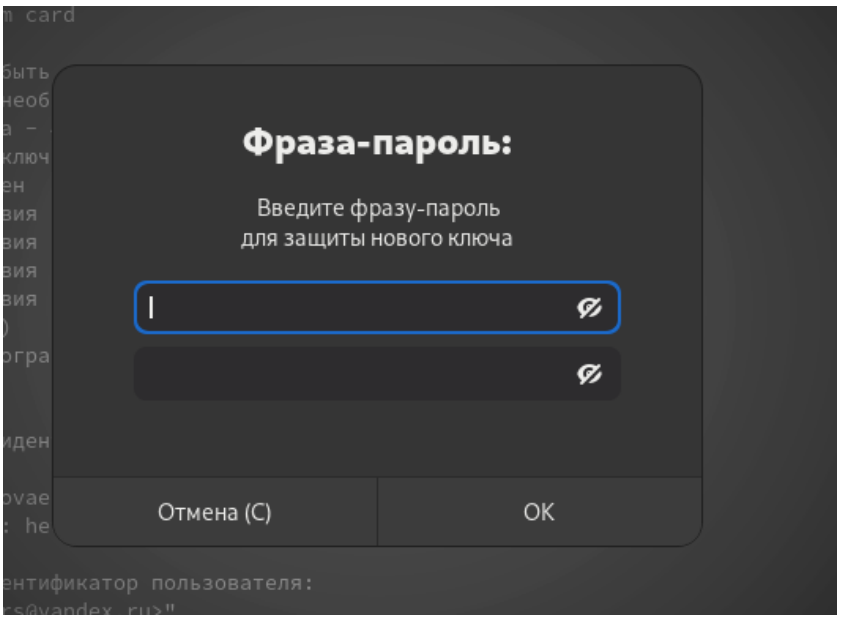

Рис. 4.15: Запрос на создание фразы-пароля

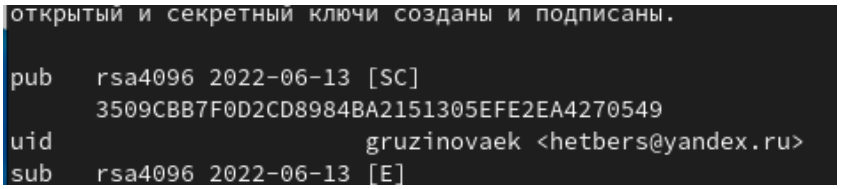

Рис. 4.16: Ключ pgp создан с необходимыми данными

8. Добавление PGP ключа в GitHub. (рис. 4.17, 4.18, 4.19, 4.20)

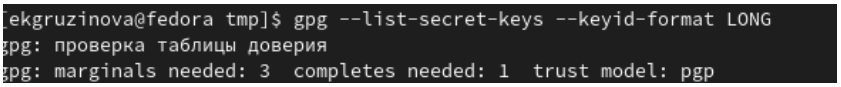

Рис. 4.17: Вывод списка ключей, из которого копирует отпечаток приватного ключа

Рис. 4.18: Копирование сгенерированного PGP ключа в буфер обмена

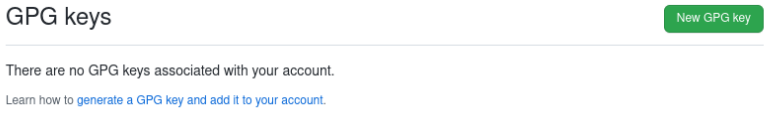

Рис. 4.19: Настройки ключа GPG на github.com

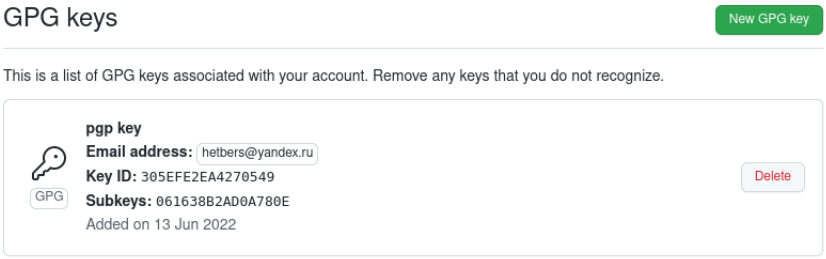

Рис. 4.20: Добавление PGP ключа в github

9. Настройка автоматических подписей коммитов git. (рис. 4.21)

```
kgruzinova@fedora tmp]$ git config --global user.signingkey 305EFE2EA4270549
kgruzinova@fedora tmp]$ git config --global commit.gpgsign true
kgruzinova@fedora tmp]$ git config --global gpg.program $(which gpg2)
```

Рис. 4.21: Через email указываем git применять его при подписи коммитов

10. Настройка gh. (рис. 4.22, 4.23, 4.24)

```
[ekgruzinova@fedora tmp]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: C7D5-B0BE
Press Enter to open github.com in your browser...
```

Рис. 4.22: Команда и процесс авторизации gh

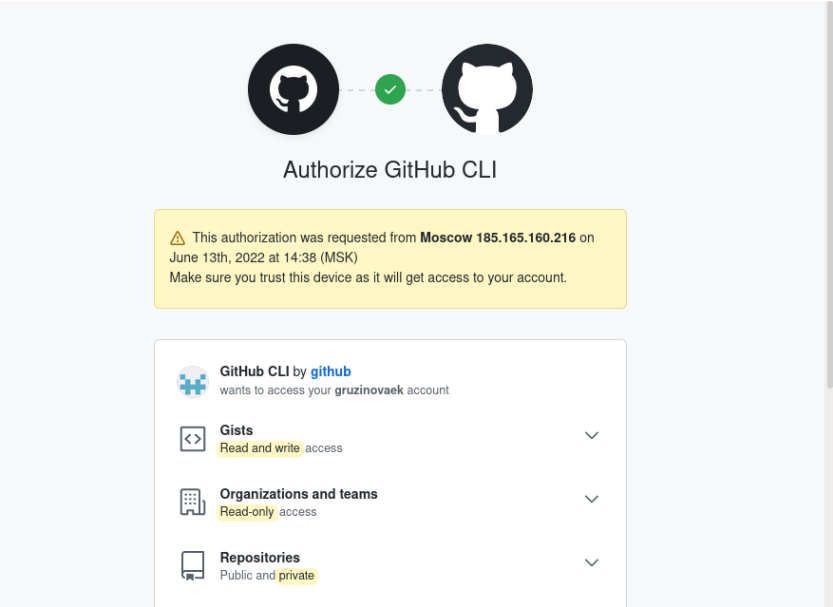

Рис. 4.23: Подтверждение авторизации происходит через браузер

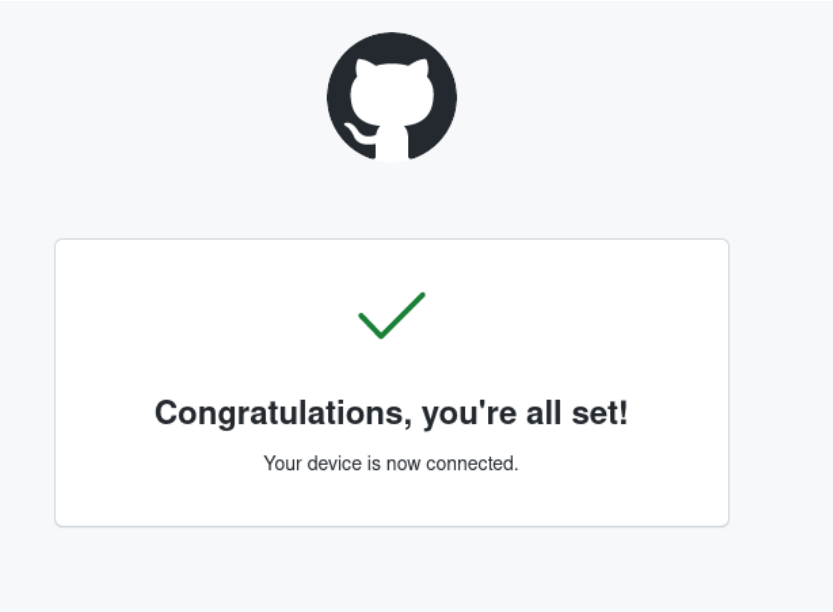

Рис. 4.24: Успешная авторизация gh

11. Создание репозитория курса на основе шаблона. (рис. 4.25, 4.26, 4.27)

```
[ekgruzinova@fedora tmp]$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
[ekgruzinova@fedora tmp]$ cd ~/work/study/2021-2022/"Операционные системы"
```

Рис. 4.25: Создание папки “Операционные системы” для репозитория

```
[ekgruzinova@fedora Операционные системы]$ gh repo create study_2021-2022_os-intro --template=yamadharma/
course-directory-student-template --public
```

Рис. 4.26: Создание репозитория на github.com

```
[ekgruzinova@fedora Операционные системы]$ git clone --recursive https://github.com/gruzinovaek/study_2021-2022_o
s-intro.git os-intro
Клонирование в «os-intro»…
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
```

Рис. 4.27: Клонирование репозитория в os-intro

12. Настройка каталога курса. (рис. 4.28, 4.29, 4.30)

```
[ekgruzinova@fedora os-intro]$ rm package.json
```

Рис. 4.28: Удаление лишних файлов

```
[ekgruzinova@fedora os-intro]$ git add .
[ekgruzinova@fedora os-intro]$ git commit -am 'feat(main): make course structure'
[master ef49fc0] feat(main): make course structure
 149 files changed, 16590 insertions(+), 14 deletions(-)
 create mode 100644 labs/lab01/presentation/Makefile
 create mode 100644 labs/lab01/presentation/presentation.md
```

Рис. 4.29: Создание необходимых каталогов

```
[ekgruzinova@fedora os-intro]$ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 266.54 КиБ | 9.87 МиБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/gruzinovaek/study_2021-2022_os-intro.git
   de2e73b..ef49fc0  master -> master
```

Рис. 4.30: Отправка файлов на сервер

5 Выводы

В процессе выполенения лабораторной работы я изучила идеологию и применение средств контроля версий, а также освоила умения по работе с git.

6 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система контроля версий — программное обеспечение, которое обеспечивает командную работу в рамках одного или нескольких проектов. Команда разработчиков взаимодействует с консольным или браузерным инструментом для выгрузки кода на сервер, скачивания его на рабочий компьютер и изменения структуры. Она хранит все версии проекта и обеспечивает к ним доступ. Любой член команды может взаимодействовать с основной «веткой» проекта или создавать новые.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. • Хранилище (repository, сокр. repo), или репозитарий, — место хранения всех версий и служебной информации. • Версия (revision), или ревизия, — состояние всех файлов на определенный момент времени, сохраненное в репозитарии, с дополнительной информацией. • Коммит (commit; редко переводится как «слепок») — • 1) синоним версии; • 2) создание новой версии («сделать коммит», «закоммитить»). • Рабочая копия (working copy или working tree) — текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней).

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специ-

альное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion. Децентрализованные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией. После внесения достаточного количества изменений в локальную копию они (изменения) отправляются на сервер. При этом сервер, чаще всего, выбирается условно, т.к. в большинстве DVCS нет такого понятия как "выделенный сервер с центральным репозиторием". Большое преимущество такого подхода заключается в автономии разработчика при работе над проектом, гибкости общей системы и повышение надежности, благодаря тому, что каждый разработчик имеет локальную копию центрального репозитория. Две наиболее известные DVCS – это Git и Mercurial.

4. Опишите действия с VCS при единоличной работе с хранилищем. Сначала вам нужно создать и подключить удаленный репозиторий. Затем, поскольку никто, кроме вас, не изменяет проект, по мере изменения проекта отправляйте изменения на сервер, и нет необходимости загружать изменения.

5. Опишите порядок работы с общим хранилищем VCS. Участник проекта (пользователь) получает нужную ему версию файлов перед началом работы с помощью определенных команд. После внесения изменений пользователь помещает новую версию в репозиторий. В то же время предыдущие версии не удаляются из центрального хранилища, и вы можете вернуться к ним в любое время.

6. Каковы основные задачи, решаемые инструментальным средством git?

У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

7. Назовите и дайте краткую характеристику командам git. git init – создание основного дерева репозитория. git pull – получение обновлений (изменений) текущего дерева из центрального репозитория. git push – отправка всех произведённых изменений локального дерева в центральный репозиторий. git status – просмотр списка изменённых файлов в текущей директории. git diff – просмотр текущих изменения. git add . – добавить все изменённые и/или созданные файлы и/или каталоги. git add имена_файлов – добавить конкретные изменённые и/или созданные файлы и/или каталоги. git rm имена_файлов – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории). git commit -am ‘Описание коммита’ – сохранить все добавленные изменения и все изменённые файлы. И т.д.

8. Что такое и зачем могут быть нужны ветви (branches)? Ветви функций, также иногда называемые ветвями тем, используются для разработки новых функций, которые должны появиться в текущих или будущих выпусках.

9. Как и зачем можно игнорировать некоторые файлы при commit? Существуют временные и системные файлы, которые загромождают проект и не нужны. путь к ним можно добавить в файл .gitignore, тогда они не будут добавлены в проект.

Список литературы

1. Лабораторная работа №2 [Электронный ресурс]. 2016. URL: https://esystem.rudn.ru/pluginfile.php/1383169/mod_resource/content/4/002-lab_vcs.pdf.