

Отчёт по лабораторной работе №2

дисциплина: Операционные системы

Латаева Гюзелия Андреевна

Содержание

1	Цель работы	5
2	Последовательность выполнения работы	6
3	Заключение	15
4	Контрольные вопросы	16

Список иллюстраций

2.1	Регистрация	6
2.2	Установка makecache	7
2.3	Установка epel_release	7
2.4	Установка gitflow	8
2.5	Установка config-manager	8
2.6	Установка репозитория с gh	8
2.7	Установка gh	8
2.8	Настройка git	9
2.9	Начальная ветка master	9
2.10	Параметры autocrlf и safecrlf	9
2.11	Алгоритм rsa	9
2.12	Алгоритм ed25519	10
2.13	Создание ключа	11
2.14	Список ключей	12
2.15	Список ключей	12
2.16	Добавление ключа GitHub	13
2.17	Настройка подписей	13
2.18	Создание репозитория	13
2.19	Копирование репозитория по шаблону	13
2.20	Переходим в каталог курса	14
2.21	Удаление лишних фалов и создание каталогов	14
2.22	Отправка данных на сервер	14

Список таблиц

1 Цель работы

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

2 Последовательность выполнения работы

1) Создаю учётную запись на <https://github.com> и заполняю данные.

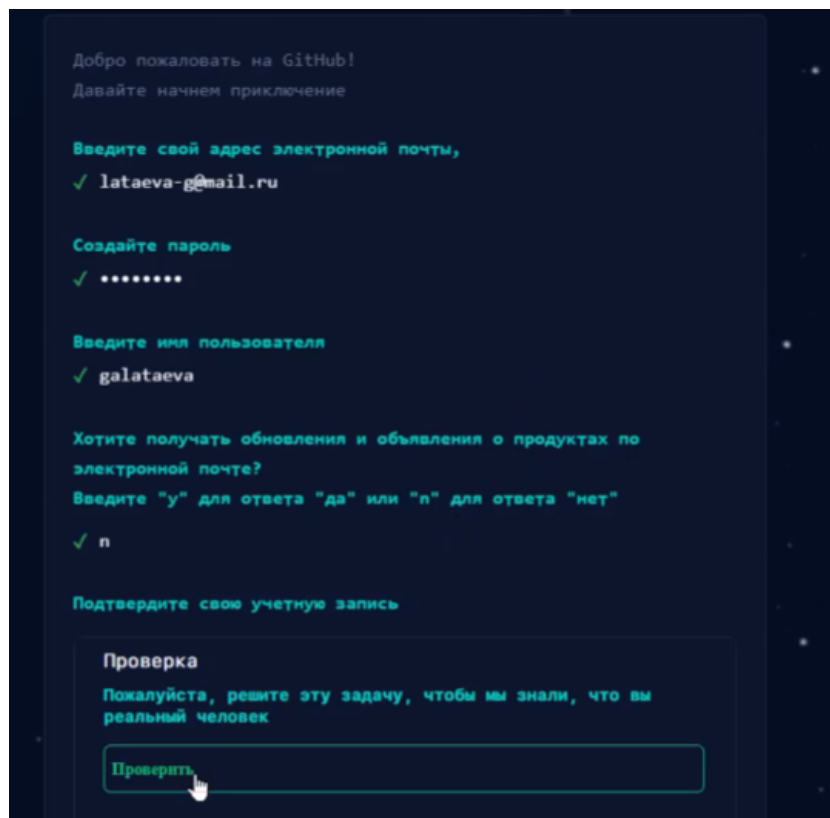
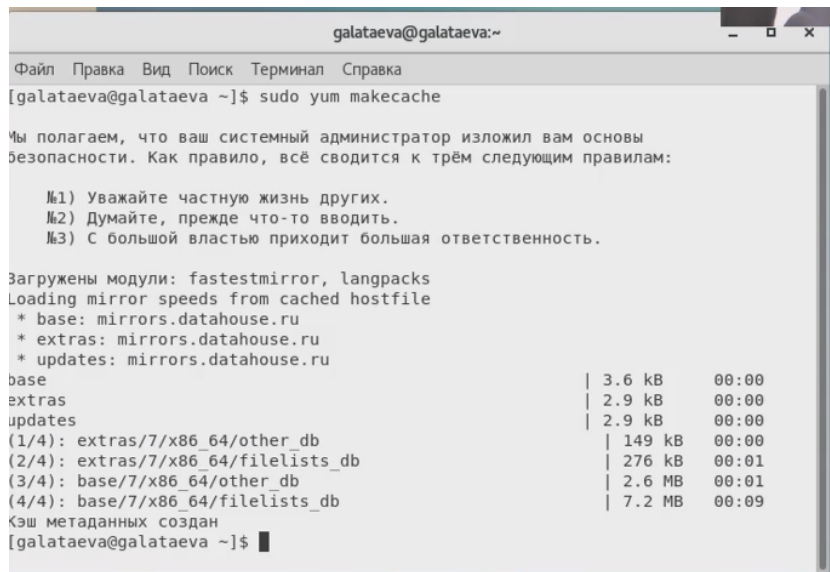


Рис. 2.1: Регистрация

2) Установка программного обеспечения

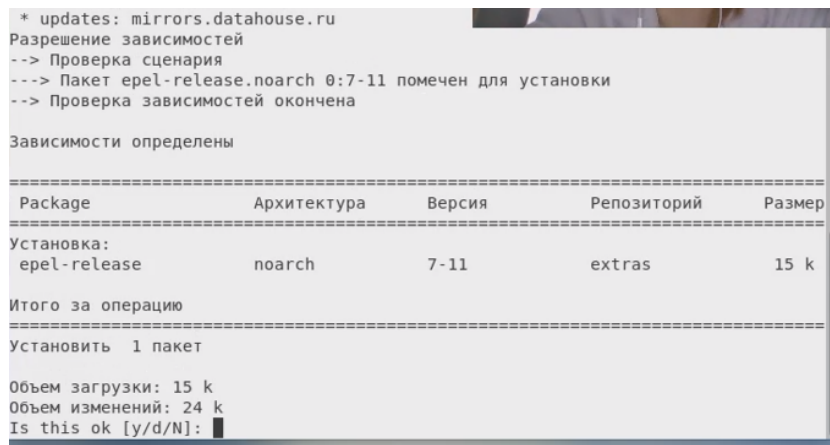
- Установка git-flow: сначала необходимо установить makescache а потом уже

сам gitflow



```
galataeva@galataeva:~  
Файл Правка Вид Поиск Терминал Справка  
[galataeva@galataeva ~]$ sudo yum makecache  
  
Мы полагаем, что ваш системный администратор изложил вам основы  
безопасности. Как правило, всё сводится к трём следующим правилам:  
  
№1) Уважайте частную жизнь других.  
№2) Думайте, прежде что-то вводить.  
№3) С большой властью приходит большая ответственность.  
  
Загружены модули: fastestmirror, langpacks  
Loading mirror speeds from cached hostfile  
* base: mirrors.datahouse.ru  
* extras: mirrors.datahouse.ru  
* updates: mirrors.datahouse.ru  
base | 3.6 kB 00:00  
extras | 2.9 kB 00:00  
updates | 2.9 kB 00:00  
(1/4): extras/7/x86_64/other_db | 149 kB 00:00  
(2/4): extras/7/x86_64/filelists_db | 276 kB 00:01  
(3/4): base/7/x86_64/other_db | 2.6 MB 00:01  
(4/4): base/7/x86_64/filelists_db | 7.2 MB 00:09  
Кэш метаданных создан  
[galataeva@galataeva ~]$
```

Рис. 2.2: Установка makescache



```
* updates: mirrors.datahouse.ru  
Разрешение зависимостей  
--> Проверка сценария  
--> Пакет epel-release.noarch 0:7-11 помечен для установки  
--> Проверка зависимостей окончена  
  
Зависимости определены  
  
=====
```

Package	Архитектура	Версия	Репозиторий	Размер
Установка: epel-release	noarch	7-11	extras	15 k

```
Итого за операцию  
=====
```

Установить	1 пакет
------------	---------

```
Объем загрузки: 15 k  
Объем изменений: 24 k  
Is this ok [y/d/N]:
```

Рис. 2.3: Установка epel_release

```
[galataeva@galataeva ~]$ sudo yum -y install gitflow
Загружены модули: fastestmirror, langpacks
Заблокировано /var/run/yum.pid: другая копия запущена с pid 28037.
Another app is currently holding the yum lock; waiting for it to exit...
Другое приложение: PackageKit
Память : 32 M RSS (604 MB VSZ)
Запущено : Sat Mar 4 17:00:47 2023 — 00:03 назад
Статус : Ожидание, pid: 28037
Another app is currently holding the yum lock; waiting for it to exit...
Другое приложение: PackageKit
Память : 32 M RSS (932 MB VSZ)
Запущено : Sat Mar 4 17:00:47 2023 — 00:05 назад
Статус : Ожидание, pid: 28037
```

Рис. 2.4: Установка gitflow

- Установка gh: на Centos 7 необходимо провести ряд манипуляций перед тем, как установить gh

```
[galataeva@galataeva ~]$ sudo dnf install gh
sudo: dnf: command not found
[galataeva@galataeva ~]$ sudo dnf install gh
sudo: dnf: command not found
[galataeva@galataeva ~]$ sudo yum install gh
Загружены модули: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
* base: mirrors.datahouse.ru
* epel: mirror.yandex.ru
* extras: mirrors.datahouse.ru
* updates: mirrors.datahouse.ru
Пакета с названием gh не найдено.
Ошибка: Выполнять нечего
[galataeva@galataeva ~]$ sudo yum install 'dnf-command(config-manager)'
Загружены модули: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
```

Рис. 2.5: Установка config-manager

```
[galataeva@galataeva ~]$ sudo yum-config-manager --add-repo https://cli.github.c
om/packages/rpm/gh-cli.repo
Загружены модули: fastestmirror, langpacks
adding repo from: https://cli.github.com/packages/rpm/gh-cli.repo
grabbing file https://cli.github.com/packages/rpm/gh-cli.repo to /etc/yum.repos.
d/gh-cli.repo
```

Рис. 2.6: Установка репозитория с gh

```
[galataeva@galataeva ~]$ sudo yum install gh
Загружены модули: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
```

Рис. 2.7: Установка gh

3) Базовая настройка git

Тут мы задали имя пользователя, почту и настроили utf-8 в выводе сообщений

```
[galataeva@galataeva ~]$ git config --global user.name "galataeva"
[galataeva@galataeva ~]$ git config --global user.email "lataeva-g@mail.ru"
[galataeva@galataeva ~]$ git config --global core.quotepath false
[galataeva@galataeva ~]$
```

Рис. 2.8: Настройка git

4) Зададим имя начальной ветки:

```
[galataeva@galataeva ~]$ git config --global init.defaultBranch master
```

Рис. 2.9: Начальная ветка master

```
[galataeva@galataeva ~]$ git config --global core.autocrlf input
[galataeva@galataeva ~]$ git config --global core.safecrlf warn
```

Рис. 2.10: Параметры autocrlf и safecrlf

5) Создаем ключи ssh

- по алгоритму rsa с ключём размером 4096 бит:

```
[galataeva@galataeva ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/galataeva/.ssh/id_rsa):
Created directory '/home/galataeva/.ssh'.

Enter same passphrase again:
Your identification has been saved in /home/galataeva/.ssh/id_rsa.
Your public key has been saved in /home/galataeva/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:0rTWa6PmJCIf4H8qED04KSRgIa9ivZE0Ww7XYknCnKE galataeva@galataeva.localdomain
The key's randomart image is:
+---[RSA 4096]-----+
|+o+oo.             |
|+o.+o o            |
|oE.+ * .           |
|*.= X .            |
|+= * .. S          |
|=. o. +            |
|o.o.. * .          |
|+.o =..o.          |
| oooooo..          |
+---[SHA256]-----+
[galataeva@galataeva ~]$
```

Рис. 2.11: Алгоритм rsa

- по алгоритму ed25519:

```
[galataeva@galataeva ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/galataeva/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/galataeva/.ssh/id_ed25519.
Your public key has been saved in /home/galataeva/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:gaUwhTg825HkayPda4RvkCS7JaQ1F76pCE0qeVY56z8 galataeva@galataeva.localdomain
The key's randomart image is:
+--[ED25519 256]--+
| . o=+ . . |
| =o+= + |
| *=0.o . |
| 0.0.X . |
| * B & o S |
| o+ 0 = . |
| . o . = |
| +E |
| .. |
+----[SHA256]-----+
[galataeva@galataeva ~]$
```

Рис. 2.12: Алгоритм ed25519

6) Создаем ключи pgr

```
[galataeva@galataeva ~]$ gpg --gen-key
gpg (GnuPG) 2.0.22; Copyright (C) 2013 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: создан каталог '/home/galataeva/.gnupg'
gpg: создан новый файл настроек '/home/galataeva/.gnupg/gpg.conf'
gpg: ВНИМАНИЕ: параметры в '/home/galataeva/.gnupg/gpg.conf' еще не активны при
этом запуске
gpg: создана таблица ключей '/home/galataeva/.gnupg/secring.gpg'
gpg: создана таблица ключей '/home/galataeva/.gnupg/pubring.gpg'
Выберите требуемый тип ключа:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (только для подписи)
  (4) RSA (только для подписи)
Ваш выбор (?-подробнее)? 1
ключи RSA могут иметь длину от 1024 до 4096 бит.
Какой размер ключа необходим? (2048) 4096
Запрашиваемый размер ключа 4096 бит
Выберите срок действия ключа.
  0 = без ограничения срока действительности
  <n> = срок действительности в днях
  <n>w = срок действительности в неделях
  <n>m = срок действительности в месяцах
  <n>y = срок действительности в годах
Ключ действителен до? (0)
Ключ не имеет ограничения срока действительности
Все верно? (y/N) y

GnuPG необходимо составить UserID в качестве идентификатора ключа.

Ваше настоящее имя: galataeva
Email-адрес: lataeva-g@mail.ru
Комментарий: hello world
Вы выбрали следующий User ID:
  "galataeva (hello world) <lataeva-g@mail.ru>"

Сменить (N)Имя, (C)Комментарий, (E)email-адрес или (O)Принять/(Q)Выход? 0
Сменить (N)Имя, (C)Комментарий, (E)email-адрес или (O)Принять/(Q)Выход? 0
Для защиты секретного ключа необходима фраза-пароль.

Необходимо сгенерировать много случайных чисел. Желательно, что бы Вы
выполняли некоторые другие активные действия (печатать на клавиатуре, движения мыш
ью,
обращения к дискам) в процессе генерации; это даст генератору
случайных чисел возможность получить лучшую энтропию.
Необходимо сгенерировать много случайных чисел. Желательно, что бы Вы
выполняли некоторые другие активные действия (печатать на клавиатуре, движения мыш
ью,
обращения к дискам) в процессе генерации; это даст генератору
случайных чисел возможность получить лучшую энтропию.
gpg: /home/galataeva/.gnupg/trustdb.gpg: создана таблица доверий
gpg: ключ 134D290B помечен как абсолютно доверяемый.
открытый и закрытый ключи созданы и подписаны.

gpg: проверка таблицы доверий
gpg: 3 ограниченных необходимо, 1 выполненных необходимо, PGP модель доверия
gpg: глубина: 0 корректных: 1 подписанных: 0 доверия: 0-, 0q, 0n, 0m, 0f,
1u
pub 4096R/134D290B 2023-03-04
Отпечаток ключа = 7432 5A2C 40B3 897A 43E7 8B9A 9A3F F9DF 134D 290B
uid galataeva (hello world) <lataeva-g@mail.ru>
sub 4096R/1713FBCD 2023-03-04
```

Рис. 2.13: Создание ключа

7) Добавляем PGP ключ в GitHub

- Выводим список ключей:

```
[galataeva@galataeva ~]$ gpg --list-secret-keys --keyid-format LONG
/home/galataeva/.gnupg/secring.gpg
-----
sec    4096R/9A3FF9DF134D290B 2023-03-04
uid          galataeva (hello world) <lataeva-g@mail.ru>
ssb    4096R/6E35AC331713FBCD 2023-03-04
```

Рис. 2.14: Список ключей

- Выводим ключ:

```
[galataeva@galataeva ~]$ gpg --armor --export 9A3FF9DF134D290B
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGQDWogBEADz1w3oI9JhRm2+4NRJXnh4kqCkD0LWJSS6lbtP50YMbn4v+Ssx
1SLMz9xrtFqTm7ieE4lMEANj+sZf0Ag52aLGi3t2Pn8i0KP0YFc71WLORIJJknqb
/aEFXaNTMDUgyb0IbXpEw8jT+iVW+AnD1jrMli0gW1v0/ohBXJ4hsImfcppT30Jt
iMgBd+RNMTRCW+fSZYm5o8wsFAWz1PHey4yTQ+c1E+nV/3mZPsPqHYvCnD9b4ln
Wpt+Phv4sNjLhJl6iclyacRgSnM9CaNuHH0YX5JuqGF0uEfPXrdi6kZhgcRVU9xm
KzvKd6RD5x2H3YE5k2J10tQS+KBRQnBBbzVio8f2Ksf0/kIzGudBYQKR+QtIbt1r
mRh5RdCavAxNuBI0nuo2fPq+ypDVoXPvon+UAKUu0wbBv/LYlohWDUCX9zIrLpqW
6jzxdSkWLF52GpxfdYhyvF9wDQ+DfEwEo/U7CBxMGB10YnY0ND8Pd9CW5CtjWWg
+YDsRYLCLSqViQL07zkIKGxcuy2eiohGNibHQYmBgjbmI1YbNK9MYDc0neRIChwB
hU89DV1MT6gyZTPn+HzwamTnXd32/4L4S0ee9Chhrey89HiybfbUamMVT9fNnAKC
XQpS3AVPyEGCL/BYfswt2S4gFxbYVpf84uB2jES2/j5i2QYfEl0IvSTUwwARAQAB
tCtnYXhhdGFldmEgKGhlbGxvIHdvcmxkKSA8bGF0YVWV2YS1nQG1haWwucnU+iQI5
BBMBAGAjBQJKA1qIAhsDBwJCAcDAgEGFQgCCQoLBBYCAwECHgECF4AAAGkQmJ/5
3xNNKQukHRAAwJa4HjQcDfJCCsEumFRvk1c6QQM1XsJYg/0QqY1nmYvVqg6Ff9tF
VAaeMgsQoXvDtZSmBrWtZ4ULR9BzQQuLx8gq4wvG+3wV227605tBrHlRo2EoJuJ
GJX8Nbziyyh/IYL67ps2uZvxBnf0kTafjbcfxZsFk05YubQnZeo4MhT3vSIBf/ek
2hSstneXhj/V20kvkBALHeg5IMJA1Yz3JUSDz27PZTayECYm82+eT9PPvS7nV1Ef
7B7o2qSsSoAUiRemi5qH0izV/rVLFu6hGYihnz5mQaoiSi3XPKHEwf3NcKtTIh5
NNq3kXNgp51YKcayHxFbluuId7nrPAbG5X/ofcbhch1JMCgu/jfUG2RyBQvGgyJN
UZHkPzSt0SY0YZNJvY8frkGKVeH5tSmIQn+Kwnld5G92qQkwp+IEE83vwNbCATtw
DliV0hxwXPR0gJxgEzWJpnRCZhlntFpQrjEsJclVVGvKTL1BMsFwgGrBRU3URpn
Hlw53qPjXLzLgSFPgUR3va81QJY/qvA0dvsI6hdouVeljvxxvJ+hifQZVdwfEdxnR
QzenGuDVFab0ugEiEl1kFwumOMrmDjYRmC3oj2kFpWNRhALMqioWEt27Iij4lyY
iRoSgXI82+4BTR+96AQD8u00oDnjRlveSshMGX1UyGzQr0cTJkCfHC65Ag0EZANa
iAEQAL7o31Eb443JHryPS24kQUC9rwSsfJzwoy2VfqCsXnyn10x0EYyX0Z+T9W2
fx7v6tNQZf+qHaKZW8AgAWfXwTw50M1FWPzR+m5+0a0qwkwxGg956xDaBfeHR83
```

Рис. 2.15: Список ключей

- Вставляем полученный ключ на сайте:

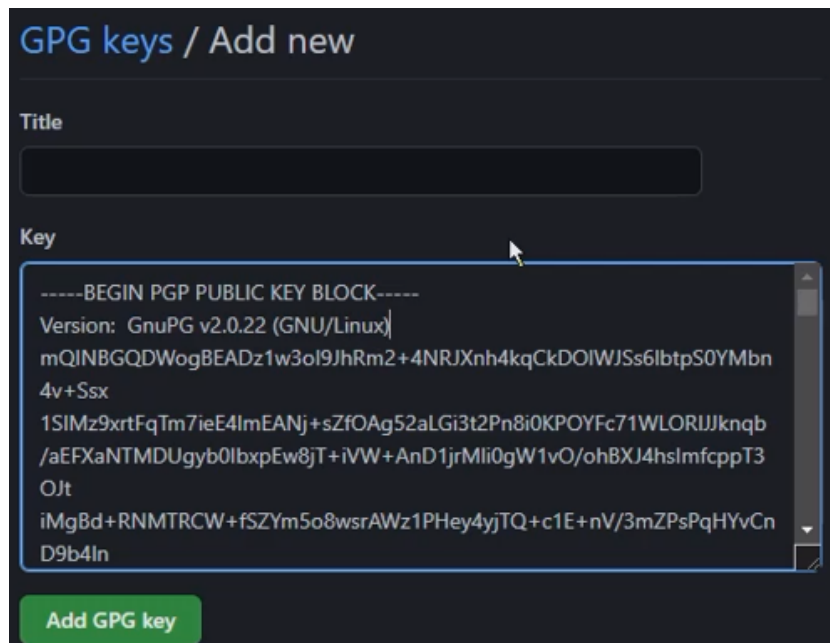


Рис. 2.16: Добавление ключа GitHub

8) Настраиваем автоматические подписи коммитов git

```
[galataeva@galataeva ~]$ git config --global user.signingkey 9A3FF9DF134D290B
[galataeva@galataeva ~]$ git config --global commit.gpgsign true
[galataeva@galataeva ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 2.17: Настройка подписей

9) Создаем репозиторий курса на основе шаблона

```
[galataeva@galataeva ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[galataeva@galataeva ~]$ cd ~/work/study/2022-2023/"Операционные системы"
[galataeva@galataeva Операционные системы]$ gh repo create study_2022-2023_os-intro --te
mplate=yamadharma/course-directory-student-template --public
✓ Created repository galataeva/study_2022-2023_os-intro on GitHub
```

Рис. 2.18: Создание репозитория

```
[galataeva@galataeva Операционные системы]$ git clone --recursive git@github.com:galataeva/study_2022-2023
_os-intro.git os-intro
Cloning into 'os-intro'...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
Receiving objects: 100% (27/27), 16.93 KiB | 0 bytes/s, done.
Resolving deltas: 100% (1/1), done.
```

Рис. 2.19: Копирование репозитория по шаблону

10) Настраиваем каталог курса

```
[galataeva@galataeva Операционные системы]$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro  
[galataeva@galataeva os-intro]$
```

Рис. 2.20: Переходим в каталог курса

```
[galataeva@galataeva os-intro]$ rm package.json  
[galataeva@galataeva os-intro]$ make COURSE=os-intro  
[galataeva@galataeva os-intro]$
```

Рис. 2.21: Удаление лишних файлов и создание каталогов

```
[galataeva@galataeva os-intro]$ git add --all package.json  
[galataeva@galataeva os-intro]$ git commit -am 'feat(main): make course structure'
```



```
[galataeva@galataeva os-intro]$ git push  
warning: push.default is unset; its implicit value is changing in  
Git 2.0 from 'matching' to 'simple'. To squelch this message  
and maintain the current behavior after the default changes, use:  
  
    git config --global push.default matching  
  
To squelch this message and adopt the new behavior now, use:  
  
    git config --global push.default simple  
  
See 'git help config' and search for 'push.default' for further information.  
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode  
'current' instead of 'simple' if you sometimes use older versions of Git)
```

Рис. 2.22: Отправка данных на сервер

3 Заключение

Я изучила применение средств контроля версий и в какой-то мере освоила навыки по работе с git.

4 Контрольные вопросы

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

Система контроля версий (от англ. Version Control System, VCS) — это, грубо говоря, место хранения кода. Она создана для разработки продуктов: на хранение кода, синхронизацию работы нескольких человек, создание релизов и т.д.

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

Хранилище (репозиторий) – это хранилище всех версий кода. Он бывает трех видов: Локальный — расположен на одном компьютере, и работать с ним может только один человек. Централизованный — расположен на сервере, куда имеют доступ сразу несколько программистов. Распределенный — самый удобный вариант с облачным хранилищем. Главный репозиторий хранится в облаке, а его локальные копии — у разработчиков на компьютерах.

Commit - запись изменений.

История - список предыдущих изменений.

Рабочая копия – текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней).

Рабочую копию получают из репозитория. Изменения вносятся в рабочую версию, потом посредством коммитов они заносятся в хранилище. История позволяет просмотреть изменения, которые вносились в репозиторий.

3. Что представляют собой и чем отличаются централизованные и

децентрализованные VCS? Приведите примеры VCS каждого вида.

См. ответ на вопрос 1.

Среди централизованных VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial.

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

- 1) Получение нужной рабочей копии
- 2) Внесение в неё необходимых изменений
- 3) Сделать нужный коммит.

5. Опишите порядок работы с общим хранилищем VCS.

См. ответ на вопрос 4 + можно объединить/отменить внесённые другими пользователями изменения, или заблокировать некоторые файлы для изменения ими.

6. Каковы основные задачи, решаемые инструментальным средством git?

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

7. Назовите и дайте краткую характеристику командам git.

git init - инициализация репозитория

git status - проверка статуса репозитория

git pull - получение обновлений (изменений)

git push - отправка изменений

git diff - просмотр текущих изменений

git add . -добавить все изменения

git add имена_файлов - добавить конкретные изменения

git rm имена_файлов - удалить файл и/или каталог из репозитория

git commit -am 'Описание коммита' - сохранение всех добавленных изменений

git log -p - просмотр истории коммитов с изменениями

`git branch` - просмотр списка веток

`git branch -d имя_ветки` - удаление ветки

`git branch имя_ветки` - создание новой ветки

`git merge имя_ветки` – слияние ветки с основной

`git merge origin` - слияние удалённого репозитория с локальным

`git push -u origin имя_ветки` - отправка новой ветки в удалённый репозиторий

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

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

Система контроля версий может поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Это удобно при работе над одним проектом нескольких человек.

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

Игнорируемые файлы отслеживаются в специальном файле `.gitignore`, который регистрируется в корневом каталоге репозитория. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо вручную отредактировать файл `.gitignore`, чтобы указать в нем новые файлы, которые должны быть проигнорированы. Файлы `.gitignore` содержат шаблоны, которые сопоставляются с именами файлов в репозитории для определения необходимости игнорировать эти файлы. Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты.