

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2дисциплина: Операционные системы

Студент: Афтаева Ксения Васильевна Преподаватель: Велиева Т.В.

Группа: НПИбд-01-20

МОСКВА 2021 г.

#####Цель работы: Изучить идеологию и применение средств контроля версий.

#####Задачи: Изучить теоретический материал. Зарегистрироваться на GitHub. Создать и настроить репозиторий. Произвести первичную конфигурацию и конфигурацию git-flow.

#####Объект и предмет исследования: GitHub

#####Техническое оснащение: Ноутбук

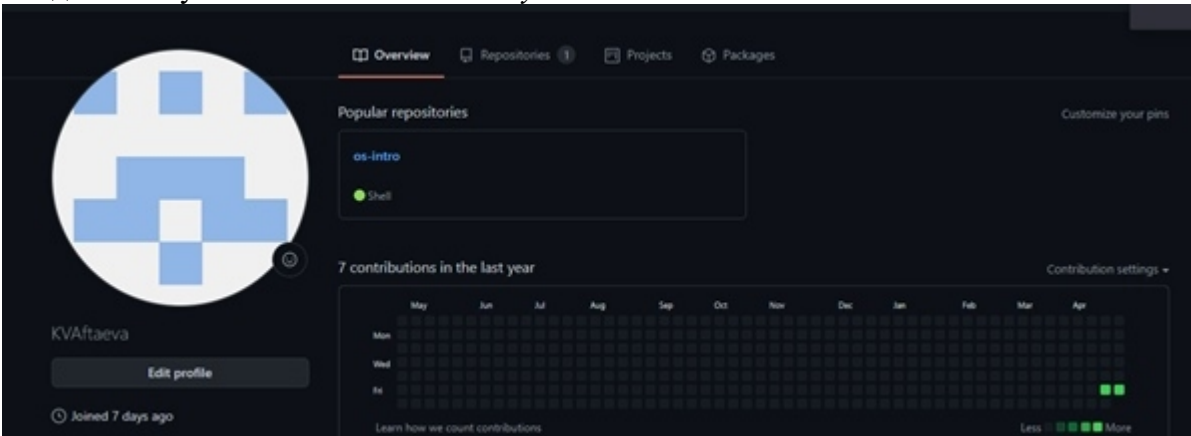
#####Теоретические вводные данные: **GitHub** — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Веб-сервис основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc (ранее Logical Awesome).

Создатели сайта называют GitHub «социальной сетью для разработчиков». Кроме размещения кода, участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых. С помощью широких возможностей Git программисты могут объединять свои репозитории — GitHub предлагает удобный интерфейс для этого и может отображать вклад каждого участника в виде дерева. Для проектов есть личные страницы, небольшие Вики и система отслеживания ошибок. Прямо на сайте можно просмотреть файлы проектов с подсветкой синтаксиса для большинства языков программирования.

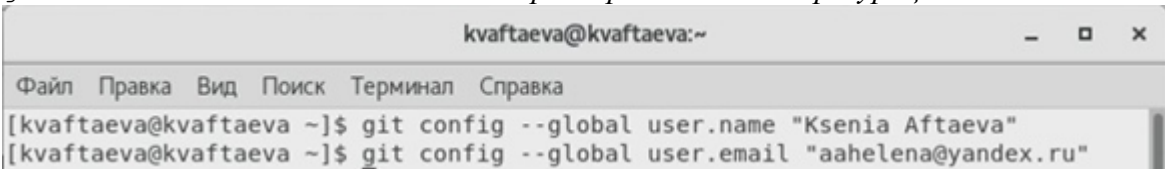
#####Условные обозначения и термины: **Репозиторий** - (от англ. repository — хранилище) — место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети.

#####Выполнение работы:

1. Создала аккаунт на GitHub *Рис.1 Аккаунт на гитхабе*



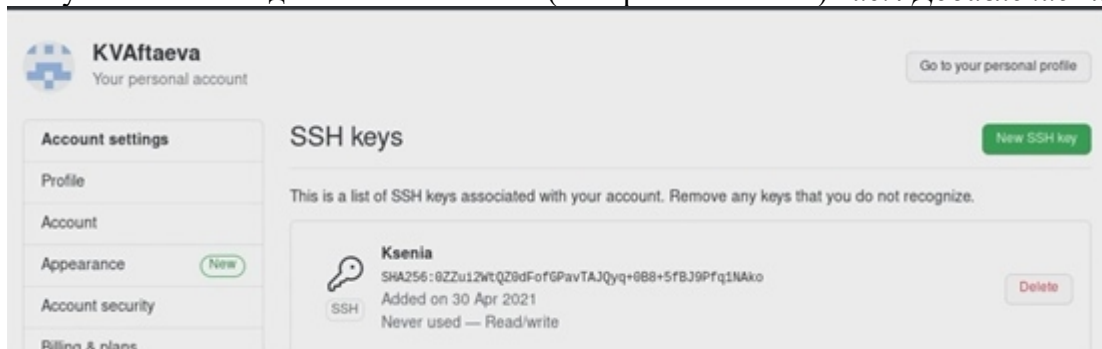
2. Создаю локальный репозиторий. Для начала делаю предварительную конфигурацию, указывая имя и почту. Происходит это при помощи команд:`git config --global user.name "Имя Фамилия"` `git config --global user.email "Почта"` *Рис.2 Предварительная конфигурация*



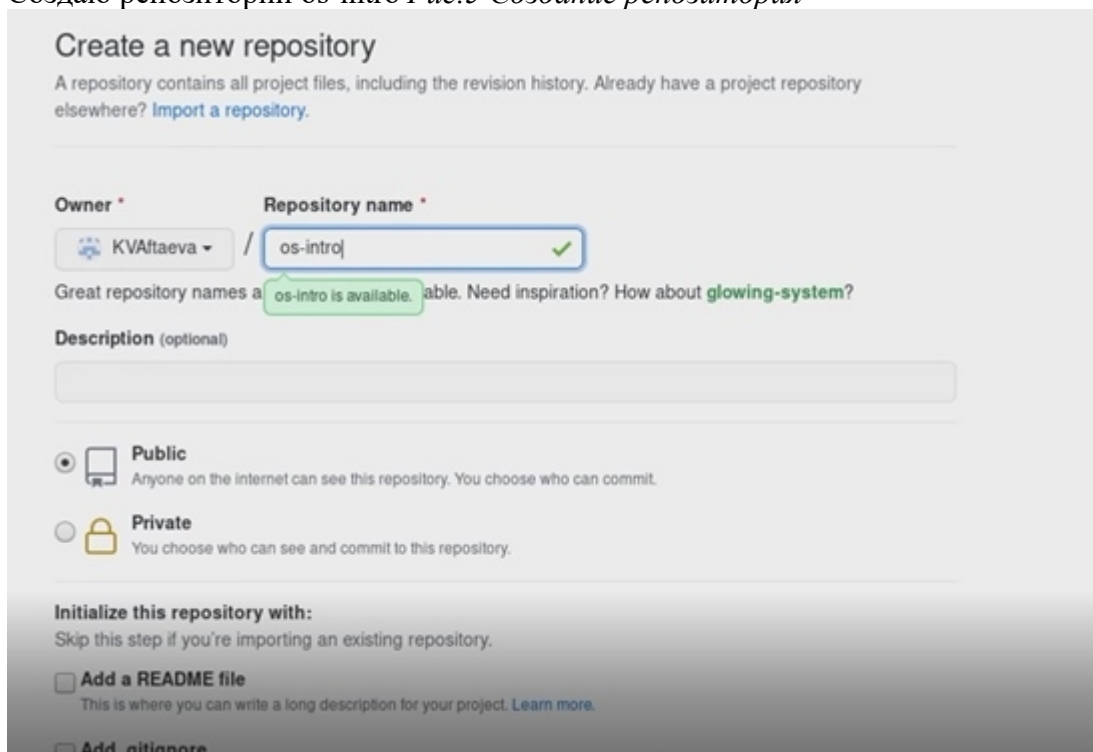
3. Для дальнейшей идентификации генерирую пару ключей с помощью команд: `*ssh-keygen -C "Имя Фамилия <Почта>"` `cat ~/.ssh/id_rsa.pub` *Рис.3 Получение ключа*

```
[kvaftaeva@kvaftaeva ~]$ ssh-keygen -C "Ksenia Aftaeva <aahelena@yandex.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kvaftaeva/.ssh/id_rsa):
/home/kvaftaeva/.ssh/id_rsa already exists.
Overwrite (y/n)?
[kvaftaeva@kvaftaeva ~]$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDS0Mo4LTQZAFrqv6cFCjzQ0eXGyk2DyIZI2XMq+1C5eCqFQHoJdQM688hV05/2IrebRBuBwoXA
JD2N0XggJ7ASp5cVgH2tx5uwiszJbMpwglBYAaMUilQwqoJF+g4V36Gf7FLDc2/aYr5PydFhN0Q8Tw4rxXRaG16YHVJwpVZPXThVAu3uSwabMuhV
shlM2w0rTYH08rLCRHQIkLezhaQ0zeTZlks0QNWjdy2d1d1lXqQuTgytc/wJ+TPX30mcZI2ra36P/6lDhduadfbcmHdw6IU436WZA6IPF1VTLVUX
fJHnukblx9AbEFL0RMx3lYmbEPEWXJWAjCeP9/g+mowb Ksenia Aftaeva <aahelena@yandex.ru>
```

#### 4. Полученный ключ добавляю на гитхаб (Настройки>Ключи) *Рис.4 Добавление ключа*



#### 5. Создаю репозиторий os-intro *Рис.5 Создание репозитория*



#### 6. Перехожу в каталог laboratory используя команду `cd <полный путь до каталога>` и инициализирую систему `git` *Рис.6 Инициализирование системы git*

```
[kvaftaeva@kvaftaeva ~]$ cd /home/kvaftaeva/work/2020-2021/os/laboratory
[kvaftaeva@kvaftaeva laboratory]$ git init
Initialized empty Git repository in /home/kvaftaeva/work/2020-2021/os/laboratory/.git/
```

#### 7. Создаю заготовку файла README.md: `echo "# Лабораторные работы" >> README.md` `git add README.md` *Рис.7 Заготовка файла README.md*

```
[kvaftaeva@kvaftaeva laboratory]$ echo "# Лабораторные работы" >> README.md
[kvaftaeva@kvaftaeva laboratory]$ git add README.md
```

#### 8. Делаю первый коммит и выкладываю на гитхаб: `git commit -m "first commit"` `git remote add origin <ссылка с гитхаба>` `git push -u origin master` *Рис.8 Первый коммит*

```
[kvaftaeva@kvaftaeva laboratory]$ git commit -m "first commit"
[master (root-commit) 7ce34f3] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
[kvaftaeva@kvaftaeva laboratory]$ git remote add origin git@github.com:KVAftaeva/os-intro.git
[kvaftaeva@kvaftaeva laboratory]$ git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 250 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:KVAftaeva/os-intro.git
* [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

На этом первая часть

работы «Подключение репозитория к гитхаб» завершена. Я создала и настроила репозиторий, загрузила туда файл.

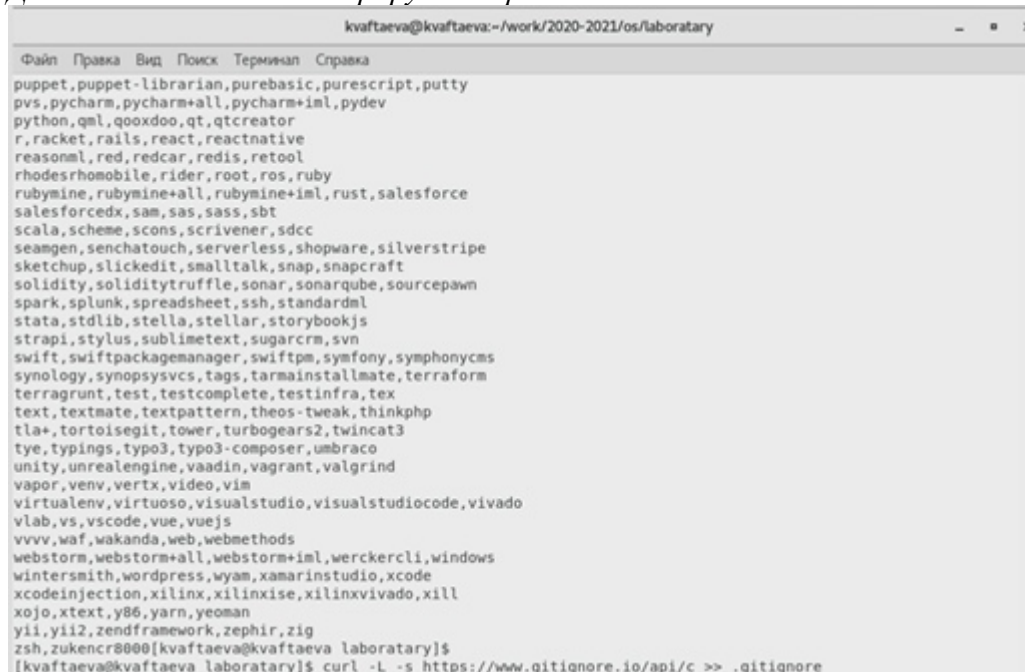
9. Добавляю файл лицензии: `wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE` *Рис.9 Добавление лицензии*

```
[kvaftaeva@kvaftaeva laboratory]$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
--2021-04-30 02:30:59-- https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)... 172.67.34.140, 104.20.151.16, 104.20.150.16, ...
Подключение к creativecommons.org (creativecommons.org)[172.67.34.140]:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENSE»

[ <=> ] 18 657 --.-K/s за 0,001s

2021-04-30 02:31:00 (22,3 MB/s) - «LICENSE» сохранён [18657]
```

10. Просматриваю список имеющихся шаблонов игнорируемых файлов и скачиваю один из них *Рис.10 Добавление шаблона игнорируемых файлов*



```
kvaftaeva@kvaftaeva:~/work/2020-2021/os/laboratory
Файл Правка Вид Поиск Терминал Справка
puppet, puppet-librarian, purebasic, purescript, putty
pvs, pycharm, pycharm+all, pycharm+iml, pydev
python, qml, qooxdoo, qt, qtcreator
r, racket, rails, react, reactnative
reasonml, red, redcar, redis, retool
rhodesrhomobile, rider, root, ros, ruby
rubymine, rubymine+all, rubymine+iml, rust, salesforce
salesforcedx, sam, sas, sass, sbt
scala, scheme, scons, scrivener, sdcc
seamgen, senchatouch, serverless, shopware, silverstripe
sketchup, slickedit, smalltalk, snap, snapcraft
solidity, soliditytruffle, sonar, sonarqube, sourcepawn
spark, splunk, spreadsheet, ssh, standardml
stata, stdlib, stella, stellar, storybookjs
strapi, stylus, sublimetext, sugarcrm, svn
swift, swiftpackagemanager, swiftpm, symfony, symphonycms
synology, synopsysvcs, tags, tarmainstallmate, terraform
terragrunt, test, testcomplete, testinfra, tex
text, textmate, textpattern, theos-tweak, thinkphp
tla+, tortoisegit, tower, turbogears2, twincat3
tye, typings, typo3, typo3-composer, umbraco
unity, unrealengine, vaadin, vagrant, valgrind
vapor, venv, vertx, video, vim
virtualenv, virtuoso, visualstudio, visualstudiocode, vivado
vlab, vs, vscode, vue, vuejs
vvvv, waf, wakanda, web, webmethods
webstorm, webstorm+all, webstorm+iml, werckercli, windows
wintersmith, wordpress, wyam, xamarinstudio, xcode
xcodeinjection, xilinx, xilinxise, xilinxvivado, xill
xojo, xtext, y86, yarn, yeoman
yii, yii2, zendframework, zephir, zig
zsh, zukencr8000[kvaftaeva@kvaftaeva laboratory]$
[kvaftaeva@kvaftaeva laboratory]$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
```

11. Добавляю новые файлы и выполняю коммит `git add .` *Рис.11 Добавление новых файлов*

```
[kvaftaeva@kvaftaeva laboratory]$ git add .
[kvaftaeva@kvaftaeva laboratory]$ git commit -a
```

В открывшемся окне при помощи клавиши ESC переходим в режим написания. Пишем текст. С помощью той же клавиши выходим из режима написания. Вводим :w для сохранения и :q для выхода

12. Отправляю на гитхаб `git push` *Рис.12 Отправка на гитхаб*

```
[kvaftaeva@kvaftaeva laboratory]$ 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)

Counting objects: 5, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 6.43 KiB | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To git@github.com:KVAftaeva/os-intro.git
 7ce34f3..341a366 master -> master
```

На этом выполнение второй

части «Первичная конфигурация закончена».

13. Ввожу дополнительные команды для установки *Рис.13 Установка*

### Other Linuxes

Under other Linuxes, the easiest way to install git-flow is using Rick Osborne's excellent git-flow installer, which can perform system-wide installation like so:

```
$ curl -OL https://raw.githubusercontent.com/nvie/gitflow/develop/contrib/gitflow-installer.sh
$ chmod +x gitflow-installer.sh
$ sudo ./gitflow-installer.sh
```

For user installation, for example in `~/bin` :

```
$ INSTALL_PREFIX=~/bin ./gitflow-installer.sh
```

14. Инициализирую git-flow `git flow init` Префикс для ярлыков устанавливаю в `v`

```
[kvaftaeva@kvaftaeva laboratory]$ git flow init

Which branch should be used for bringing forth production releases?
- master
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] v
```

*Рис.14 Инициализирование и*

*установка префикса*

15. Проверяю, что нахожусь на ветке `develop` `git branch` *Рис.15 Проверка ветки*

```
[kvaftaeva@kvaftaeva laboratory]$ git branch
* develop
master
```

16. Создаю релиз с версией `1.0.0` `git flow release start 1.0.0` И записываю версию `echo "1.0.0" >> VERSION` *Рис.16 Создание релиза и запись версии*



```
[kvaftaeva@kvaftaeva laboratory]$ git flow release start 1.0.0
Switched to a new branch 'release/1.0.0'

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish '1.0.0'

[kvaftaeva@kvaftaeva laboratory]$ echo "1.0.0" >> VERSION
```

17. Добавляю в индекс: `git add .` `git commit -am 'chore(main): add version'` *Рис.17 Добавление в*

*индекс*

```
[kvaftaeva@kvaftaeva laboratory]$ git add .
[kvaftaeva@kvaftaeva laboratory]$ git commit -am 'chore(main): add version'
[release/1.0.0 4138283] chore(main): add version
3 files changed, 80 insertions(+)
create mode 100644 VERSION
create mode 100000 gitflow
create mode 100755 gitflow-installer.sh
[kvaftaeva@kvaftaeva laboratory]$
```

18. Зальём релизную ветку в основную ветку `git flow release finish 1.0.0` *Рис.18 Релизную ветку в*

*основную*

```
create mode 100755 gitflow-installer.sh
[kvaftaeva@kvaftaeva laboratory]$ git flow release finish 1.0.0
```

В

открывшемся окне при помощи клавиши ESC переходим в режим написания. Пишем текст. С помощью той же клавиши выходим из режима написания. Вводим :w для сохранения и :q для выхода

19. Отправим данные и создадим релиз на гитхабе *Рис.19 Отправка данных*

```
[kvaftaeva@kvaftaeva laboratory]$ git push --all
Counting objects: 7, done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 1.38 KiB | 0 bytes/s, done.
Total 6 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To git@github.com:KVAftaeva/os-intro.git
 341a366..857c49f master -> master
 * [new branch]      develop -> develop
[kvaftaeva@kvaftaeva laboratory]$ git push --tags
Counting objects: 1, done.
Writing objects: 100% (1/1), 161 bytes | 0 bytes/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To git@github.com:KVAftaeva/os-intro.git
 * [new tag]          v1.0.0 -> v1.0.0
[kvaftaeva@kvaftaeva laboratory]$ git push --follow-tags
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)
```

#####Контрольные вопросы :

- Системы контроля версий (VCS)** - программное обеспечение для облегчения работы с изменяющейся информацией, позволяющее хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям. Предназначены для работы нескольких человек над одним проектом, а также при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы.
- Хранилище** – место «памяти», в котором будет храниться новая версия файла после его изменения пользователем. Commit. В нем содержится описание тех изменений, которые вносит пользователь в код приложения. История – история изменений. Обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Рабочая копия – это копия, которую мы выписали в свою рабочую зону, это то, над чем мы работаем в данный момент. Привилегированный доступ только одному пользователю, работающему с файлом.

3. Централизованные VCS предполагают наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Пример: AscuRev  
Децентрализованные VCS не имеют единого репозитория, он у каждого пользователя свой. Помимо того, они были созданы для обмена изменениями, а не для их объединения. Не имеют какой-то жестко заданной структуры репозитория с центральным сервером. Пример: Git
4. При единоличной работе с VCS каждое новое изменение в репозитории сохраняется не со всеми предыдущими версиями. Оно изменяется по системе: одно предыдущее + новая информация.
5. Для начала те действия, что совершаются один раз:
  - Создать репозиторий.
  - Это место, где будут лежать файлы. Теперь у нас есть общее хранилище данных, с которым и будет проходить дальнейшая работа.
  - Скачать проект из репозитория.

Далее то, что будет использоваться в работе часто:

- Забрать последнюю версию
  - Внести изменения в проект
  - Запустить код, т.е изменить код в общем хранилище
  - Создать ветку
  - Теперь, если нужно закоммитить изменения, они по-прежнему пойдут в основную ветку. Бранч при этом трогать НЕ будут. Так что мы можем смело коммитить новый код в trunk. А для показа использовать branch, который будет оставаться стабильным даже тогда, когда в основной ветке всё падает из-за кучи ошибок. С бранчами мы всегда будем иметь работающий код.
6.
    - Сохранение файлов с исходным кодом
    - Защита от случайных исправлений и удалений
    - Отмена изменений и удалений, если они оказались некорректными
    - Одновременная поддержка рабочей версии и разработка новой
    - Возврат к любой версии кода из прошлого
    - Просмотр истории изменений
    - Совместная работа без боязни потерять данные или затереть чужую работу
  7. **Локальный репозиторий** – она же директория “.git”. В ней хранятся коммиты и другие объекты. Удаленный репозиторий – тот репозиторий, который считается общим, в который мы можем передать свои коммиты из локального репозитория, чтобы остальные пользователи могли их увидеть. Локальный репозиторий мы используем, когда работаем одни и нам нужно сохранить свои же изменения. Удаленный репозиторий используется для групповой работы, когда в личном репозитории скопилось достаточно коммитов, мы делимся ими в удаленном для того, чтобы другие пользователи могли видеть наши изменения. Также из удаленного репозитория мы можем скачать чужие изменения.
  8. **Ветка** – это подвижный указатель на один из коммитов. Обычно ветка указывает на последний коммит в цепочке коммитов. В своей ветке мы можем как угодно ломать проект, основной код при этом не пострадает.
  9. **Игнорируемые файлы** – это, как правило, специфичные для платформы файлы или автоматически созданные файлы из систем сборки. Некоторые общие примеры включают в себя:
    - Файлы времени выполнения, такие как журнал, блокировка, кэш или временные файлы.
    - Файлы с конфиденциальной информацией, такой как пароли или ключи API.
    - Скомпилированный код, такой как .class или .o.
    - Каталоги зависимостей, такие как /vendor или /node\_modules.
    - Создавать папки, такие как /public, /out или /dist.
    - Системные файлы, такие как .DS\_Store или Thumbs.db
    - Конфигурационные файлы IDE или текстового редактора.

**.gitignore Шаблоны** .gitignore — это простой текстовый файл, в каждой строке которого содержится шаблон, который файлы или каталоги следует игнорировать. Он использует шаблоны подстановки для сопоставления имен файлов с подстановочными знаками. Если у вас есть файлы или каталоги, содержащие шаблон

подстановки, вы можете использовать одиночную обратную косую черту ( ) для экранирования символа.

**Местный .gitignore** .gitignore файл .gitignore обычно помещается в корневой каталог репозитория. Однако вы можете создать несколько файлов .gitignore в разных подкаталогах вашего репозитория. Шаблоны в файлах .gitignore сопоставляются относительно каталога, в котором находится файл. Шаблоны, определенные в файлах, которые находятся в каталогах (подкаталогах) более низкого уровня, имеют приоритет над шаблонами в каталогах более высокого уровня. Локальные файлы .gitignore используются совместно с другими разработчиками и должны содержать шаблоны, полезные для всех других пользователей репозитория. **Личные правила игнорирования** Шаблоны, специфичные для вашего локального репозитория и не подлежащие распространению в другие репозитории, должны быть установлены в файле .git/info/exclude . Например, вы можете использовать этот файл, чтобы игнорировать файлы, сгенерированные из ваших личных инструментов проекта. **Глобальный .gitignore** Git также позволяет вам создать глобальный файл .gitignore , в котором вы можете определить правила игнорирования для каждого репозитория Git в вашей локальной системе. Файл можно назвать как угодно и хранить в любом месте. Чаще всего этот файл хранится в домашнем каталоге. Вам придется вручную создать файл и настроить Git для его использования.

#####Заключение: Таким образом, все цели и задачи были выполнены успешно. Был освоен GitHub, который понадобится в дальнейшей работе.

#####Вывод: Я изучила идеологию и применение средств контроля версий.

#####Библиографический список:

1. A bright future for GitHub | The GitHub Blog
2. <https://gist.github.com/rdnvndr/cb21a06c5a71fd71213aed1619380b8e>