

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

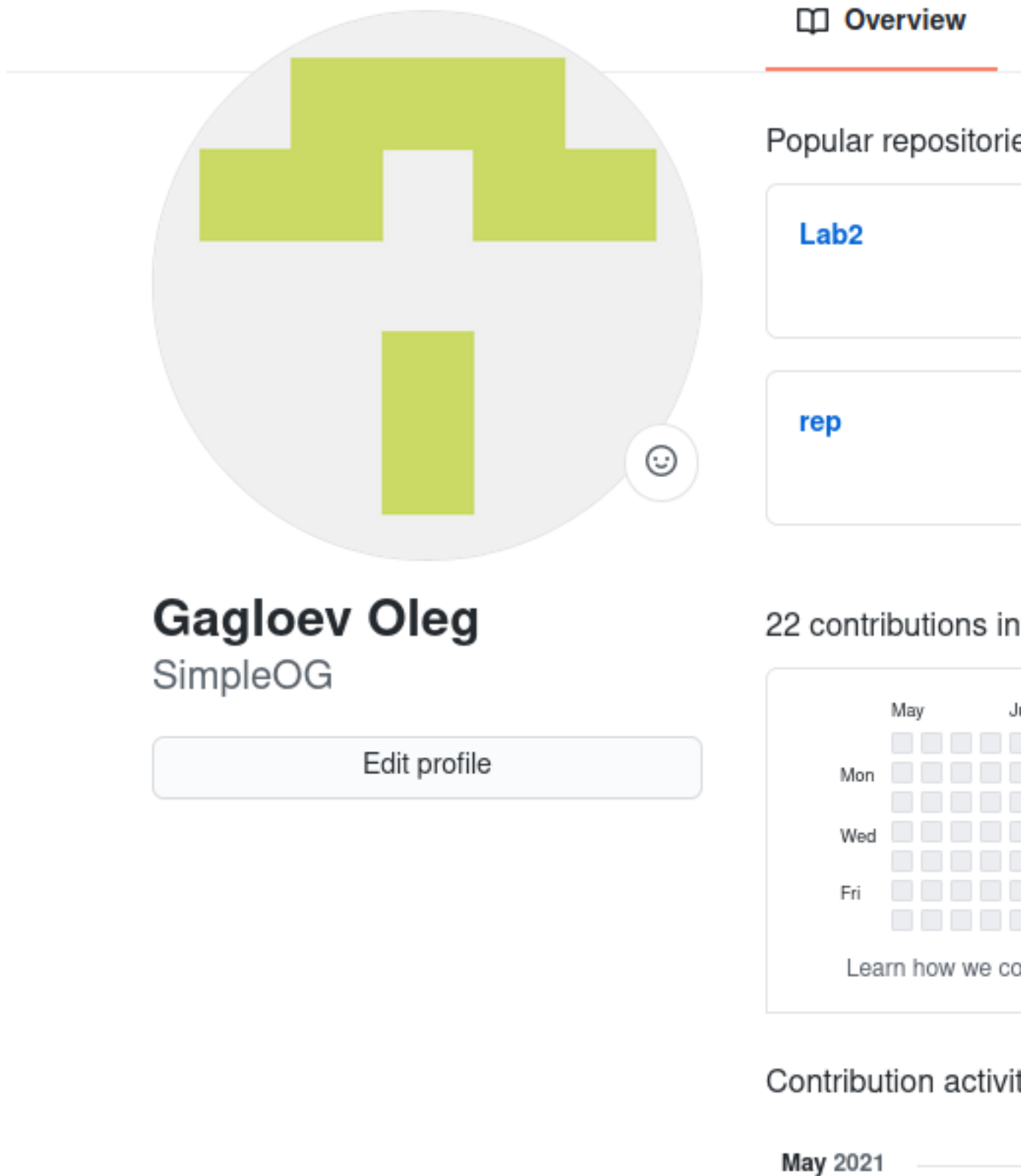
Кафедра прикладной информатики и теории вероятностей

Отчёт по лабораторной работе №2 ##### Тема : Управление версиями Выполнил:  
Студент группы НПИ 01-20 Гаглов Олег Мелорович Студенческий билет №  
1032201347

Москва 2021 ##### Структура : 1. Цель 2. Ход работы 3. Вывод ##### Цель : Изучить идеологию и применение средств контроля версий.

*Ход работы :*

1) Создал репозиторий на github <https://github.com/SimpleOG/rep.git>



The screenshot shows a GitHub profile page for a user named Gagloev Oleg. The profile picture is a circular avatar with a light gray background and a green geometric design. Below the avatar, the name "Gagloev Oleg" is displayed in a large, bold font, followed by the username "SimpleOG" in a smaller font. A button labeled "Edit profile" is located below the name. To the right of the profile information, there is a section titled "Overview" with a book icon. Below this, the "Popular repositories" section lists two repositories: "Lab2" and "rep". The "rep" repository is highlighted. Below the repositories, it shows "22 contributions in" followed by a calendar view for May. The calendar shows a grid of days with some days highlighted in gray. Below the calendar, there is a link "Learn how we co". At the bottom of the page, the "Contribution activity" section is visible, showing a bar chart for May 2021.

**Overview**

Popular repositories

**Lab2**

**rep**

22 contributions in

May

Mon

Wed

Fri

Learn how we co

Contribution activity

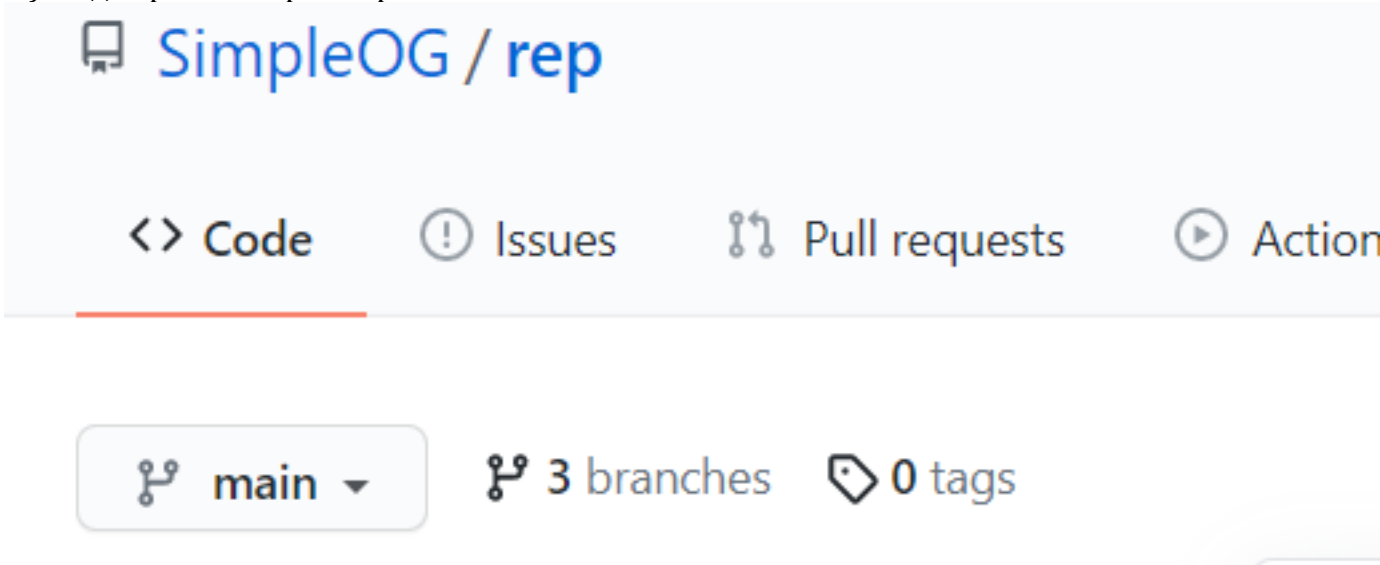
May 2021

2)Получил ssh ключ с помощью команды ssh-keygen -C "Имя Фамилия " И

скопировал его с помощью команды `cat ~/.ssh/id_rsa.pub | xclip -sel clip`, после чего установил его на github

```
om-gagloev@omgagloev:~$ ssh-keygen -C "Oleg Gagloev <gagloev@gmail.com>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/om-gagloev/.ssh/id_rsa):
/home/om-gagloev/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/om-gagloev/.ssh/id_rsa
Your public key has been saved in /home/om-gagloev/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:UV+2CYJFZhpsRr2vqE6QCP3od/5Gq6fNIbftKImdf68 Oleg Gagloev <gagloev@gmail.com>
The key's randomart image is:
+---[RSA 3072]---+
|           oo=B . o           |
| .           =*.o + o         |
|. .         oo  .. o          |
| . + .      ..                |
| o +   S    .                 |
| . . . . .                    |
| . ..++oo .                   |
| ..+===+o                     |
|   o*XB+Eo.                   |
+-----[SHA256]-----+
om-gagloev@omgagloev:~$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
om-gagloev@omgagloev:~$
```

3) Создал репозиторий rep



2) Создал папку lab02.С помощью следующих команд загрузил из локального каталога на сервер репозиторий

`git remote add origin ssh://git@github.com//.git`  
`git push -u origin master`  
5) В папке lab02 начал производить действия с файлами:

```

om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/Лабораторные работы">> README.md
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/Лабораторные работы">> README.md
Инициализирован пустой репозиторий Git в /home/om-gagloev/work/2020-2021/Операционные системы/lab02/.git/
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/Лабораторные работы">> README.md
.md
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/Лабораторные работы">> README.md
"first commit"
[master (корневой коммит) 15e49d4] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/Лабораторные работы">> README.md
main
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/Лабораторные работы">> README.md
origin git@github.com:SimpleOG/rep.git
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/Лабораторные работы">> README.md
igin main
Warning: Permanently added the RSA host key for IP address '192.168.1.100' to the
list of known hosts.
Перечисление объектов: 3, готово.
Подсчет объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 248 bytes | 248.00 KiB/s, готово.
Всего 3 (изменения 0), повторно использовано 0 (изменения 0)
To github.com:SimpleOG/rep.git
* [new branch]      main -> main
Ветка «main» отслеживает внешнюю ветку «main» из «origin».

```

Инициализировал систему гит, добавил заготовку для файла README.md (echo "# Лабораторные работы" >> README.md и -git add README.md). Сделал первый коммит и выложил на гитхаб с помощью команд -git commit -m "first commit" -git remote add origin git@github.com:/sciproc-intro.git -git push -u origin master 6) Добавил файл лицензии с помощью команды

wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE

После чего добавил шаблон игнорируемых файлов. Просмотрев список имеющихся шаблонов: с помощью команды curl -L -s https://www.gitignore.io/api/list

```
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/Lab02$ curl -L -s https://www.gitignore.io/api/list
```

```
1c,1c-bitrix,a-frame,actionscript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,ansibletower,apachecordova
apachehadoop,appbuilder,appcelerator titanium,appcode,appcode+iml,appengine,aptanastudio,arcanist,archive
archives,archlinuxpackages,aspnetcore,assembler,ate
atmelstudio,ats,audio,automationstudio,autotools
autotools+strict,awr,azurefunctions,backup,ballerina
basercms,basic,batch,bazaar,bazel
bitrise,bitrix,bittorrent,blackbox,bloop
bluej,bookdown,bower,bricxcc,buck
c,c++,cake,cakephp,cakephp2
cakephp3,calabash,carthage,certificates,ceylon
cfwheels,chefcookbook,chocolatey,clean,clion
clion+all,clion+iml,clojure,cloud9,cmake
cocoapods,cocos2dx,cocoscreator,code-java,codeblocks
codecomposerstudio,codeigniter,codeio,codekit,codesniffer
```

Затем скачиваю шаблон, например, для C: с помощью команды `curl -L -s https://www.gitignore.io/api/c >> .gitignore` и добавляю новые файлы с помощью `git add .`

```
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/Lab02$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/Lab02$ git add .
```

Далее создаем первый коммит с помощью команды `git commit -a`



И отправлю на github с помощью команды git push

```
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/lab02$ git commit -a
На ветке main
Ваша ветка опережает «origin/main» на 1 коммит.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)

ничего коммитить, нет изменений в рабочем каталоге
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/lab02$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (4/4), 6.45 KiB | 6.45 MiB/s, готово.
Всего 4 (изменения 0), повторно использовано 0 (изменения 0)
```

7) Приступлю к конфигурации gitflow

Инициализировал git flow и посмотрел на какой ветке нахожусь

```
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/lab02$ git flow init
Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [/home/om-gagloev/work/2020-2021/Операционные системы/lab02/.git/hooks]
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/lab02$ git flow init
* develop
  main
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/lab02$ git flow set versiontagprefix v
```

Ветка develop Так же установил префикс для ярлыков в v Создам релиз с помощью команды git flow release start 1.0.0



Запишу его версию с помощью команды echo

```
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/
ase start 1.0.0
Переключено на новую ветку «release/1.0.0»

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develo
- 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 relea
- When done, run:

    git flow release finish '1.0.0'

om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/
VERSION
echo1.0.0: команда не найдена
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/
> VERSION
```

Добавим в индекс текст с помощью команд git add . git commit -am 'chore(main): add version'

```
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/lab02$ git add .
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/lab02$ git commit -a
m'chore(main): add version'
[release/1.0.0 727021b] chore(main): add version
 1 file changed, 1 insertion(+)
 create mode 100644 VERSION
```

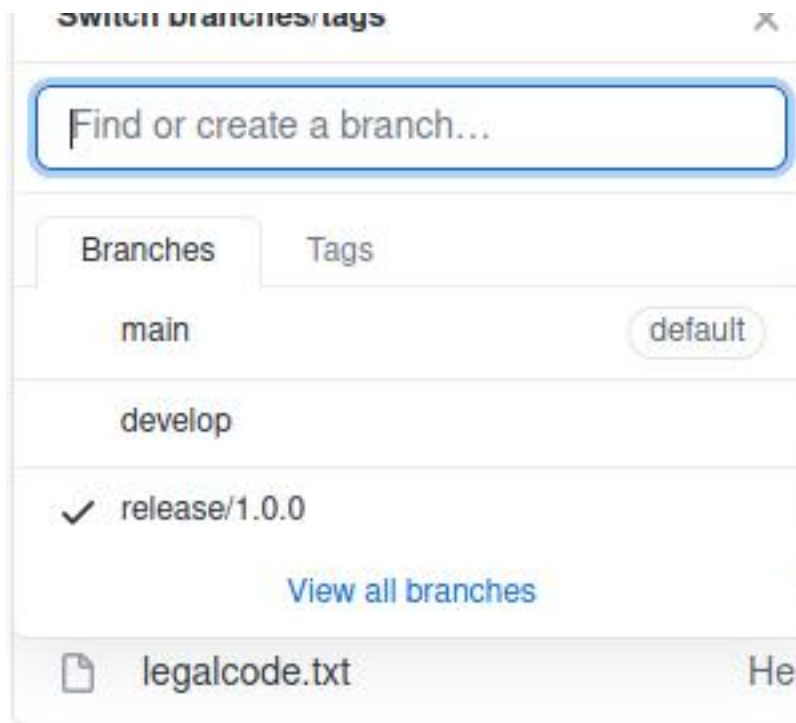
Зальём релизную ветку в основную ветку git flow release finish 1.0.0

```
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/1
ase finish 1.0.0
Переключено на ветку «main»
Ваша ветка обновлена в соответствии с «origin/main».
Merge made by the 'recursive' strategy.
  VERSION | 1 +
  1 file changed, 1 insertion(+)
  create mode 100644 VERSION
Уже на «main»
Ваша ветка опережает «origin/main» на 2 коммита.
  (используйте «git push», чтобы опубликовать ваши локальные
fatal: нет описания метки?
Fatal: Tagging failed. Please run finish again to retry.
```

Отправим данные на github git push --all git push --tags

```
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/1
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
Сжатие объектов: 100% (3/3), готово.
Запись объектов: 100% (4/4), 384 bytes | 384.00 KiB/s, готово
Всего 4 (изменения 2), повторно использовано 0 (изменения 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local
To github.com:SimpleOG/rep.git
  00f192c..d43cccf  main -> main
* [new branch]      develop -> develop
* [new branch]      release/1.0.0 -> release/1.0.0
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/1
gs
Everything up-to-date
om-gagloev@omgagloev:~/work/2020-2021/Операционные системы/1
```

Создадим релиз на github



### Вывод Я изучил идеологию и применение средств контроля, научился работать с репозиторием через терминал

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.

class=WordSection2>

. Централизованные VCS предполагают наличие единого репозитория для хранения файлов. Выполнение

. большинства функций по управлению версиями осуществляется специальным сервером. Пример: AccuRev

. 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

. При единоличной работе с VCS каждое новое изменение в репозитории сохраняется не со всеми

. предыдущими версиями. Оно изменяется по системе: одно предыдущее + новая информация.

. 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 закомитить 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.

- . ◦ 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

. **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.

. **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.

. **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.