

Отчёт лабораторной работы № 3

Markdown

Кекишева Анастасия Дмитриевна

Содержание

1	Цель работы	5
2	Задание к лабораторной работе №2	6
3	Выполнение лабораторной работы	7
4	Вывод:	12

Список таблиц

Список иллюстраций

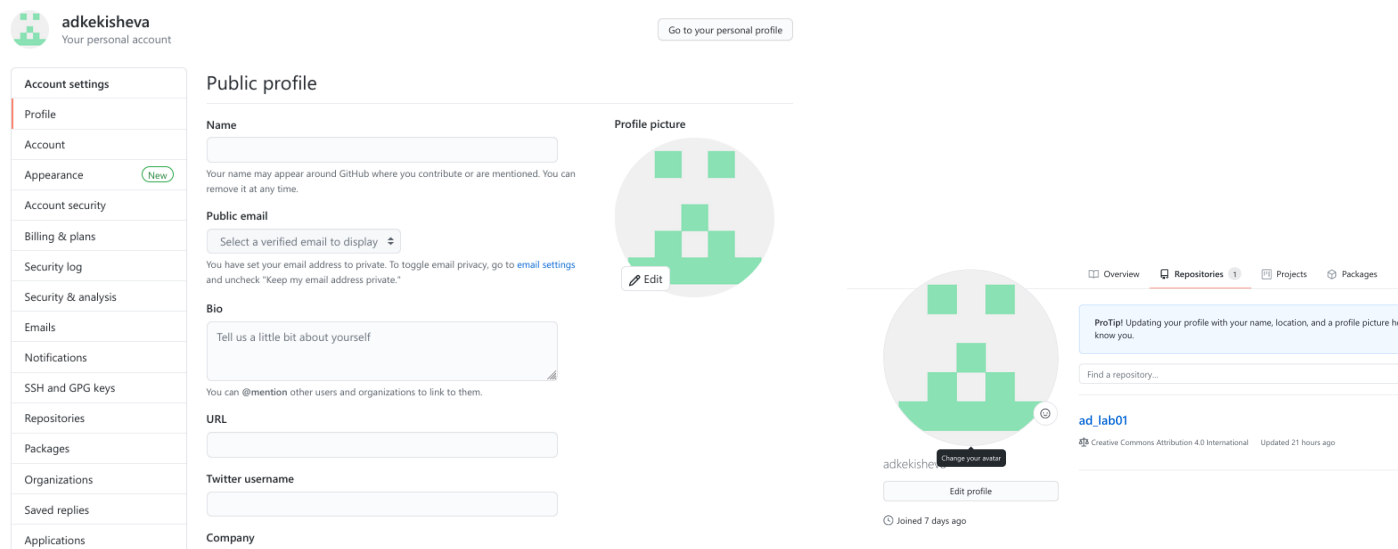
1 Цель работы

Изучить идеологию и применение средств контроля версий.

2 Задание к лабораторной работе №2

- Настройка системы Git;
- Подключение репозитория к github;
- Первичная конфигурация, добавление файла лицензии;
- Конфигурация git-flow;

3 Выполнение лабораторной работы



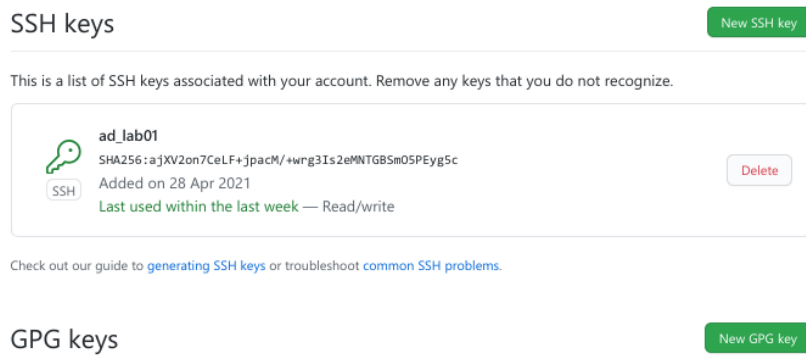
Прежде всего, я создала аккаунт на github, а также создала пустой репозиторий.

```
adkekisheva@dk8n78:~/laboratory
adkekisheva@dk8n78 ~$ cd .ssh
adkekisheva@dk8n78 ~/.ssh$ ssh-keygen -C "Анастасия"
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/a/d/adkekisheva/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/a/d/adkekisheva/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/a/d/adkekisheva/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:uoXs+5U0QGT20Ur3KAYrM501a1/EsqBXkP9HEFdbwY Анастасия
The key's randomart image is:
+---[RSA 3072]-----+
|      ..=o.oE.o|
|      .+. = oooo|
|      ...o=o.o.+|
|      . . +.+.o |
|      o .Soo. o .|
|      .+o..=o. o |
|      o+...oo= . |
|      ..o..o o .|
|      +o.  o    |
+---[SHA256]-----+
```

Далее, перешла в папку .ssh и в ней сгенерировала пароль с помощью команды `ssh-keygen-C`

```
+-----[SHA256]-----+
adkeki sheva@dk8n78 ~/.ssh $ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCfXysRviCDJibT3DMeGpKAERDayuk1b6zERhB++QRXlk7zj15zQ7Tiz7mgvBRa2zI
5uMqcfv+Aqr/77dQh+rzgCChs14EF3DHf2+hS0s3L9F/7MggsmWLAtnZMuXY/FmdAg80JqWDFf2c0ARu19zd6hTdpw28YK5mMb043RB
G4Y0KN6U6UEb+I5azM2iQLtX4K67ybElrYAh6IGU02MOzQ34QmUTgFWMMtDubOKoFvhhMZ1U/u6nkYcIV1NKHLLQD3sPBqLjQvXjeu
06jbxLwY2/MS841VV12rysoNmb1cqfYrZzcYrRvOgfsShX5JxuBLficqBRueY/GRIIXzk30Xyo06URroKfwtHrxMxGmnivMS8Hgkxm
LoTy8CAetUy2U3ntL0yeX//6Exh+xGuS8Py4IafhstYhx1LCabaYaMCBebBzP5ldWdIYNn5emT+DReJySGcd+3MFwK1M4LJXzDwZ8wZ
P40yp/rxV//HFif0= Анастасия
adkeki sheva@dk8n78 ~/.ssh $ cat id_rsa.pub | xclip -sel clip
```

После с помощью команды `cat id_rsa.pub` просмотрела публичный вопрос а затем командой `cat id_rsa.pub | xclip -sel clip` скопировала этот пароль.



Вставила скопированный публичный пароль, дала название ключу, и добавила SSH ключ на github.

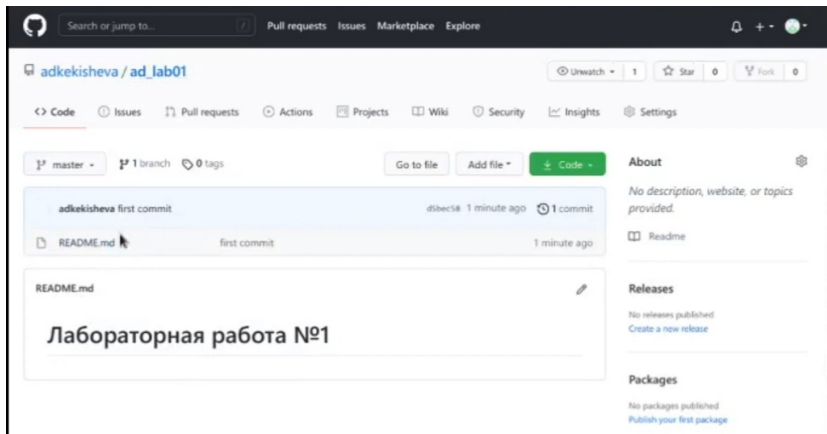
```
adkeki sheva@dk8n78 ~/.ssh $ cat id_rsa.pub | xclip -sel clip
adkeki sheva@dk8n78 ~/.ssh $ cd
adkeki sheva@dk8n78 ~ $ mkdir laboratory
mkdir: невозможно создать каталог «laboratory»: Файл существует
adkeki sheva@dk8n78 ~ $ cd laboratory
adkeki sheva@dk8n78 ~/laboratory $ git init
Инициализирован пустой репозиторий Git в /afs/.dk.sci.pfu.edu.ru/home/a/d/adkeki sheva/laboratory/.git/
adkeki sheva@dk8n78 ~/laboratory $ echo "# Лабораторная работа №1" >> README.md
adkeki sheva@dk8n78 ~/laboratory $ git add README.md
adkeki sheva@dk8n78 ~/laboratory $ git commit -am "first commit"
```

Перешла в домашний каталог и создала в нём папку `laboratory` (`mkdir laboratory`) и перешла в неё. Далее инициализировала `git`-репозиторий (`git init`) и создала заготовку для файла `README.md`: в этот файл я добавила строку «Лабораторная работа №1», а потом с помощью команды `git add` добавила файл `README.md` и сохранила все изменения в папке с помощью команды `git commit -am` с описанием «first commit».

```
adkeki sheva@dk8n78 ~/laboratory $ git remote add origin git@github.com:adkeki sheva/ad_lab01.git
adkeki sheva@dk8n78 ~/laboratory $ git push -u origin master
The authenticity of host 'github.com (140.82.121.3)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com,140.82.121.3' (RSA) to the list of known hosts.
Перечисление объектов: 100% (3/3), готово.
Подсчет объектов: 100% (3/3), 246 bytes | 246.00 KiB/s, готово.
Запись объектов: 100% (3/3), 246 bytes | 246.00 KiB/s, готово.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

To github.com:adkeki sheva/ad_lab01.git
 * [new branch] master -> master
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
```

И чтобы закинуть файл `README.md` на github я воспользовалась командами `git add origin` + ссылка на репозиторий и `git push -u`.



Проверила репозиторий: у меня создался файл README.mdi в нём появился заголовок «Лабораторная работаNo1».

```
adkeisheva@dk8n78 ~/laboratory $ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
--2021-04-28 11:09:41-- https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)... 104.20.150.16, 172.67.34.140, 104.20.151.16, ...
Подключение к creativecommons.org (creativecommons.org)[104.20.150.16]:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENSE»

LICENSE [ <=> ] 18,22K --.-KB/s за 0,001s

2021-04-28 11:09:41 (13,4 MB/s) - «LICENSE» сохранён [18657]
```

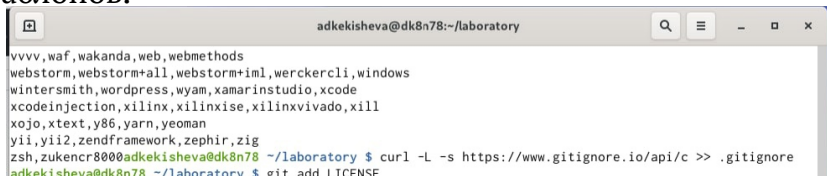
Добавила файл лицензии.

```
LICENSE [ <=> ] 18,22K --.-KB/s за 0,001s

2021-04-28 11:09:41 (13,4 MB/s) - «LICENSE» сохранён [18657]

adkeisheva@dk8n78 ~/laboratory $ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionsript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,apachecordova,apachehadoop
appbuilder,appcelerator titanium,appcode,appcode+all,appcode+iml
```

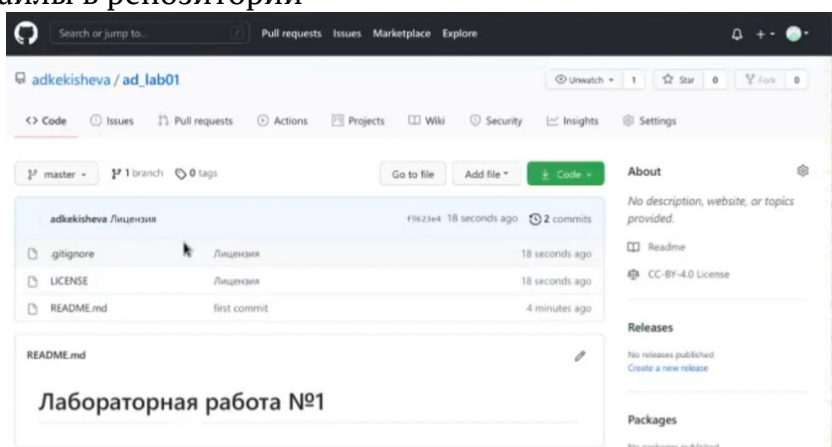
Добавила шаблон игнорируемых файлов и просмотрела список имеющихся шаблонов.



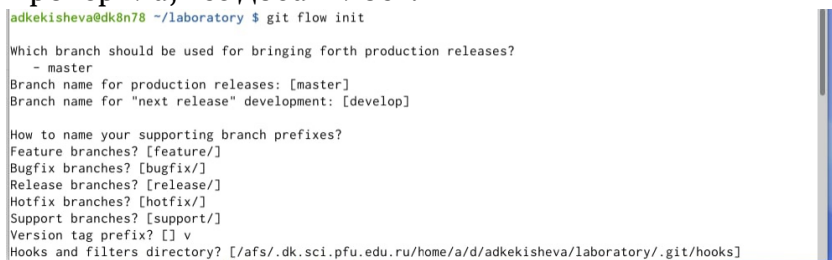
И скачала шаблон

```
adkeisheva@dk8n78 ~/laboratory $ git add LICENSE
adkeisheva@dk8n78 ~/laboratory $ git add .gitignore
adkeisheva@dk8n78 ~/laboratory $ git commit -am "Лицензия"
[master f9623e4] Лицензия
2 files changed, 455 insertions(+)
create mode 100644 .gitignore
create mode 100644 LICENSE
adkeisheva@dk8n78 ~/laboratory $ git push
Warning: Permanently added the RSA host key for IP address '140.82.121.4' to the list of known hosts.
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (4/4), 6.45 KiB | 6.45 MiB/s, готово.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:adkeisheva/ad_lab01.git
d5bec58..f9623e4 master -> master
```

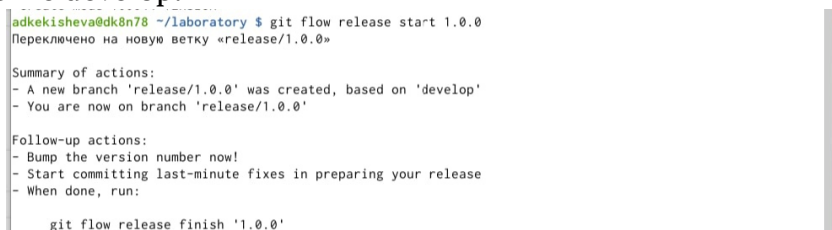
Добавила и сохранила файлы LICENSE и .gitignore в текущем каталоге, потом сохранила все добавленные изменения с описанием (Лицензия). И отправила файлы в репозиторий



Проверила, всё добавилось.



Проинициализировала git-flow командой git-flow init. Префикс для ярлыков установила в v. Далее, с помощью команды git branch проверила, что нахожусь в ветке develop.



Создала релиз с версией 1.0.0

```
adkekisheva@dk8n78 ~/laboratory $ echo " 1.0.0" >> VERSION
adkekisheva@dk8n78 ~/laboratory $ git add VERSION
adkekisheva@dk8n78 ~/laboratory $ git commit -am 'chore(main): add version'
[release/1.0.0 06f8f6e] chore(main): add version
1 file changed, 1 insertion(+)
adkekisheva@dk8n78 ~/laboratory $ git flow release finish 1.0.0
```

Записала в файл VERSION строку с версией, добавила и сохранила изменения.

```
adkekisheva@dk8n78 ~/laboratory $ git flow release finish 1.0.0
Переключено на ветку «master»
Ваша ветка обновлена в соответствии с «origin/master».
Merge made by the 'recursive' strategy.
VERSION | 2 ++
1 file changed, 2 insertions(+)
create mode 100644 VERSION
Уже на «master»
Ваша ветка опережает «origin/master» на 3 коммита.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)
Переключено на ветку «develop»
Merge made by the 'recursive' strategy.
VERSION | 1 +
1 file changed, 1 insertion(+)
Ветка release/1.0.0 удалена (была 06f8f6e).

Summary of actions:
- Release branch 'release/1.0.0' has been merged into 'master'
- The release was tagged 'v1.0.0'
- Release tag 'v1.0.0' has been back-merged into 'develop'
- Release branch 'release/1.0.0' has been locally deleted
- You are now on branch 'develop'
```

Залила релизную ветку в основную ветку.

```
adkekisheva@dk8n78 ~/laboratory $ git push --all
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (8/8), 806 bytes | 806.00 KiB/s, готово.
Total 8 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To github.com:adkekisheva/ad_lab01.git
 f9623e4..3e44450 master -> master
 * [new branch]      develop -> develop
git adkekisheva@dk8n78 ~/laboratory $ git push --tags
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 165 bytes | 165.00 KiB/s, готово.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:adkekisheva/ad_lab01.git
 * [new tag]         v1.0.0 -> v1.0.0
adkekisheva@dk8n78 ~/laboratory $
```

И отправила все фали и теги на github.

The screenshot shows a GitHub repository interface. At the top, it says 'adkekisheva Merge branch "release/1.0.0"' with a commit hash '36c645a' and '21 hours ago'. Below this is a table of files: .gitignore, LICENSE, README.md, and VERSION, each with a commit message and time. The README.md file is expanded, showing the text 'Лабораторная работа №1'. On the right side, there is a 'Releases' section showing 'v1.0.0' as the latest release, with a 'Draft a new release' button at the top right.

Проверила, у меня создались 2 ветки, а также 1 тег.

The screenshot shows the 'Releases' page for the repository. It displays the 'v1.0.0' release, which was released '21 hours ago'. Below the release information, there is a section for 'Assets' with two items: 'Source code (zip)' and 'Source code (tar.gz)'. At the top right of the page, there is a 'Draft a new release' button.

И создала релиз v1.0.0

4 Вывод:

Я познакомилась с системой контроля версий Git, изучила основные команды для работы с git, получила практические навыки по работе с сервером репозитория, а также, как вносить изменения, делать коммиты и выкладывать их на github, также поработала с gitflow, научилась создавать релиз с версией, заливать этот релиз в основную ветку и создавать релиз на github.

Контрольные вопросы:

1. Система управления версиями-программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. В основном применяются при работе нескольких человек над одним проектом.
2. Хранилище или репозиторий-место хранения всех версий и служебной информации. Commit - процесс создания новой версии. После внесения изменений пользователь может сохранить все добавленные изменения и все изменённые файлы. Понятие истории в vcs можно объяснить так: в любой момент пользователь может вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения, а также просмотреть истории изменений. Рабочая копия-текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней).
3. Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и-

хранится на сервере. Доступ к нему осуществляется через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion. Распределенные (децентрализованные) системы контроля версий 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. Две наиболее известные DVCS – это Git и Mercurial.

4. При единоличной работе с хранилищем, как правило, есть репозиторий и несколько копий файла, над которым идёт работа. Все изменения сохраняются, и можно вернуться в любому этапу работы, а также объединять файлы.
5. Участник проекта после внесения изменений, 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. Можно объединить изменения, сделанные разными участниками, вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом и отправлять все изменения в хранилище.

6. 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. Команды:

- `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;
- `git checkout -b имя_ветки` - создание новой ветки, базирующейся на текущей;
- `git checkout` имя_ветки -переключение на некоторую ветку(при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой);
- `git push origin имя_ветки` -отправка изменений конкретной ветки в центральный репозиторий;
- `git merge --no-ff имя_ветки` -слияние ветки с текущим деревом;
- `git branch -d имя_ветки` -удаление локальной уже слитой с основным деревом ветки;
- `git branch -D имя_ветки` -принудительное удаление локальной ветки;

8. Локальный репозиторий хранится на нашем компьютере, в рабочей папке проекта, в которой мы проинициализировали репозиторий и в которой находится папка `.git`, в которой будет храниться история изменений. Удалённый репозиторий хранится в облаке, на сторонних сервисах, специально созданных под работу с проектами `git`. Удаленный репозиторий выполняет роль резервной копии, даёт возможность работать в команде, а также некоторые дополнительные возможности. Например, визуализация истории или возможность работать над проектом прямо в веб-интерфейсе.
9. Для фиксации истории проекта в рамках этого процесса вместо одной ветки `master` используются две ветки. В ветке `master` хранится официальная история релиза, а ветка `develop` предназначена для объединения всех функций. Помимо главных ветвей `master` и `develop`, наша модель разработки содержит некоторое количество типов вспомогательных ветвей, которые используются для распараллеливания разработки между членами команды, для упрощения внедрения нового функционала (`features`), для подготовки релизов и для быстрого исправления проблем в производственной версии приложения. В целом ветки позволяют решать задачи система управления версиями `git`.
10. Во время работы над проектом так или иначе могут создаваться файлы (временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами), которые не требуется добавлять в последствии в репозиторий, поэтому мы можем их игнорировать. Чтобы устранить ненужные файлы, можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов. Для этого сначала нужно получить список имеющихся шаблонов, затем скачать шаблон, например, для C