

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1.2
дисциплины «Информационные технологии и программирование»

Выполнила:
Мурашко Анастасия Юрьевна
1 курс, группа ИТС-б-0-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Братченко Н.Ю., канд. физ.-мат. наук,
доцент, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Исследование основных возможностей Giti GitHub.

Цель: исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

Конспект теоретического материала:

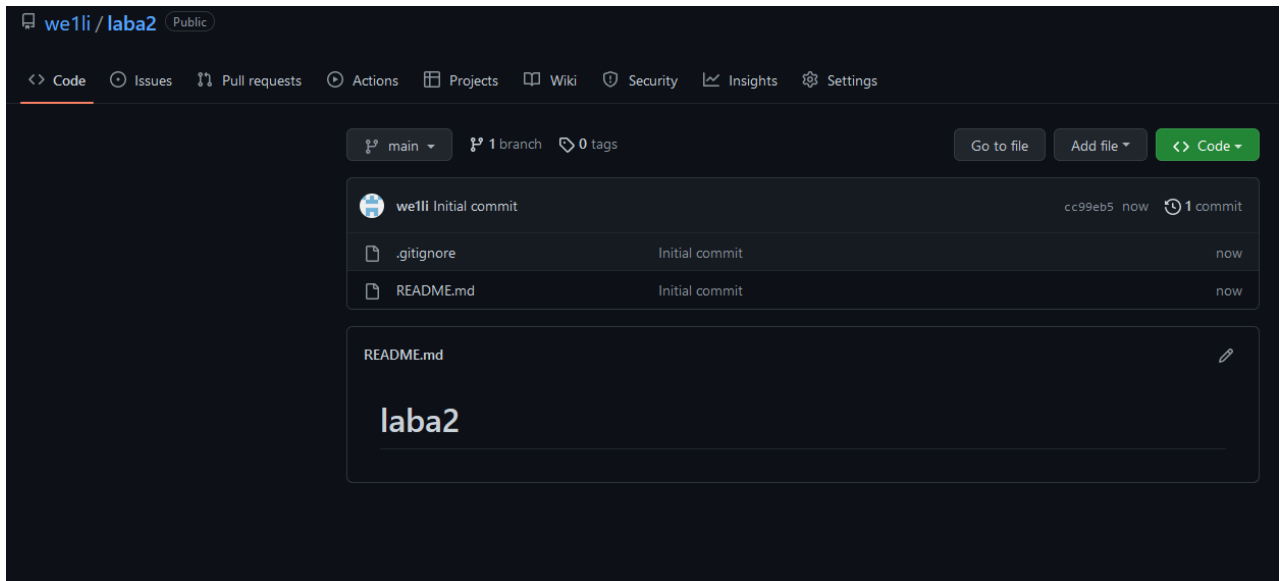
После того, как вы создали несколько коммитов или же клонировали репозиторий с уже существующей историей коммитов, вероятно Вам понадобится возможность посмотреть что было сделано — историю коммитов. Одним из основных и наиболее мощных инструментов для этого является команда `git log`. По умолчанию (без аргументов) `git log` перечисляет коммиты, сделанные в репозитории в обратном к хронологическому порядку — последние коммиты находятся вверху. Из примера можно увидеть, что данная команда перечисляет коммиты с их SHA-1 контрольными суммами, именем и электронной почтой автора, датой создания и сообщением коммита. Команда `git log` имеет очень большое количество опций для поиска коммитов по разным критериям. Рассмотрим наиболее популярные из них.

Одним из самых полезных аргументов является `-p` или `--patch`, который показывает разницу (выводит патч), внесенную в каждый коммит. Так же вы можете ограничить количество записей в выводе команды; используйте параметр `-2` для вывода только двух записей.

Следующей действительно полезной опцией является `--pretty`. Эта опция меняет формат вывода. Существует несколько встроенных вариантов отображения. Опция `oneline` выводит каждый коммит в одну строку, что может быть очень удобным если вы просматриваете большое количество коммитов. К тому же, опции `short`, `full` и `fuller` делают вывод приблизительно в том же формате, но с меньшим или большим количеством информации.

Порядок выполнения работы:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования.



2. Проработайте примеры лабораторной работы. Отрадите вывод на консоли при выполнении команд git в отчете для лабораторной работы.

```
C:\Users\user\Documents\GitHub>git clone https://github.com/we1li/laba2.git
Cloning into 'laba2'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Комманда «git clone [https://...](https://github.com/we1li/laba2.git)»

```
C:\Users\user\Documents\GitHub\laba2>git log
commit cc99eb5862fc23292f2a2c59d8865e8ebc527ec0 (HEAD -> main, origin/main, origin/HEAD)
Author: we1li <125406661+we1li@users.noreply.github.com>
Date: Tue Mar 14 21:05:15 2023 +0300

Initial commit
```

Комманда «git log».

```
diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..b6e4761
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1,129 @@
+# Byte-compiled / optimized / DLL files
+__pycache__/
+*.py[cod]
+*.py.class
+
+# C extensions
+*.so
+
+# Distribution / packaging
+.Python
+build/
+develop-eggs/
+dist/
+downloads/
+eggs/
+*.eggs/
+lib/
```

Комманда «git log -p -2»

Отмена индексации файла

```
C:\Users\user\Documents\GitHub\laba2>git reset head 26_1.py
```

Комманда «git reset HEAD»

Отмена изменений в файле

```
C:\Users\user\Documents\GitHub\laba2>git checkout -- 26_1.py  
C:\Users\user\Documents\GitHub\laba2>git status  
On branch main  
Your branch is ahead of 'origin/main' by 3 commits.  
  (use "git push" to publish your local commits)  
  
nothing to commit, working tree clean
```

Комманда «git checkout --».

Просмотр удалённых репозиториев

```
C:\Users\user\Documents\GitHub\laba2>git remote -v  
origin  https://github.com/welli/laba2.git (fetch)  
origin  https://github.com/welli/laba2.git (push)
```

Комманда «git remote -v».

Добавление удалённых репозиториев

```
C:\Users\user\Documents\GitHub\laba2>git remote add pb https://github.com/welli/laba2.git  
C:\Users\user\Documents\GitHub\laba2>git remote -v  
origin  https://github.com/welli/laba2.git (fetch)  
origin  https://github.com/welli/laba2.git (push)  
pb      https://github.com/welli/laba2.git (fetch)  
pb      https://github.com/welli/laba2.git (push)  
C:\Users\user\Documents\GitHub\laba2>git fetch pb  
From https://github.com/welli/laba2  
* [new branch]      main -> pb/main
```

Комманда «git fetch»

Просмотр удаленного репозитория

```
C:\Users\user\Documents\GitHub\laba2>git remote show origin  
* remote origin  
Fetch URL: https://github.com/welli/laba2.git  
Push URL: https://github.com/welli/laba2.git  
HEAD branch: main  
Remote branch:  
  main tracked  
Local branch configured for 'git pull':  
  main merges with remote main  
Local ref configured for 'git push':  
  main pushes to main (fast-forwardable)
```

Комманда «git remote show origin».

Удаление и переименование удалённых репозиторий

```
C:\Users\user\Documents\GitHub\laba2>git remote rename pb paul  
Renaming remote references: 100% (1/1), done.
```

Комманда «git remote rename».

```
C:\Users\user\Documents\GitHub\laba2>git remote remove paul  
  
C:\Users\user\Documents\GitHub\laba2>remote  
"remote" не является внутренней или внешней  
командой, исполняемой программой или пакетным файлом.  
  
C:\Users\user\Documents\GitHub\laba2>git remote
```

Комманда «git remote remove paul».

Работа с тегами

Просмотр списка тегов

```
C:\Users\user\Documents\GitHub\laba2>git tag
```

Комманда «git tag»

Создание тегов.

```
C:\Users\user\Documents\GitHub\laba2>git tag -a v1.4 -m "my version 1.4"
```

Комманда «git tag -a»

Аннотированные теги

```
C:\Users\user\Documents\GitHub\laba2>git show v1.4  
tag v1.4  
Tagger: Nastya <nastya.murashko1@yandex.ru>  
Date: Tue Mar 14 22:59:11 2023 +0300  
  
my version 1.4  
  
commit 822f4eb8f469f49517dda53d44057eb7ed75a11c (HEAD -> main, tag: v1.4)  
Author: Nastya <nastya.murashko1@yandex.ru>  
Date: Tue Mar 14 22:35:46 2023 +0300  
  
    gitignore  
  
diff --git a/.gitignore b/.gitignore  
index b6e4761..aa44ee2 100644  
--- a/.gitignore  
+++ b/.gitignore  
@@ -127,3 +127,4 @@ dmyru.json  
  
# Pyre type checker  
.pyre/
```

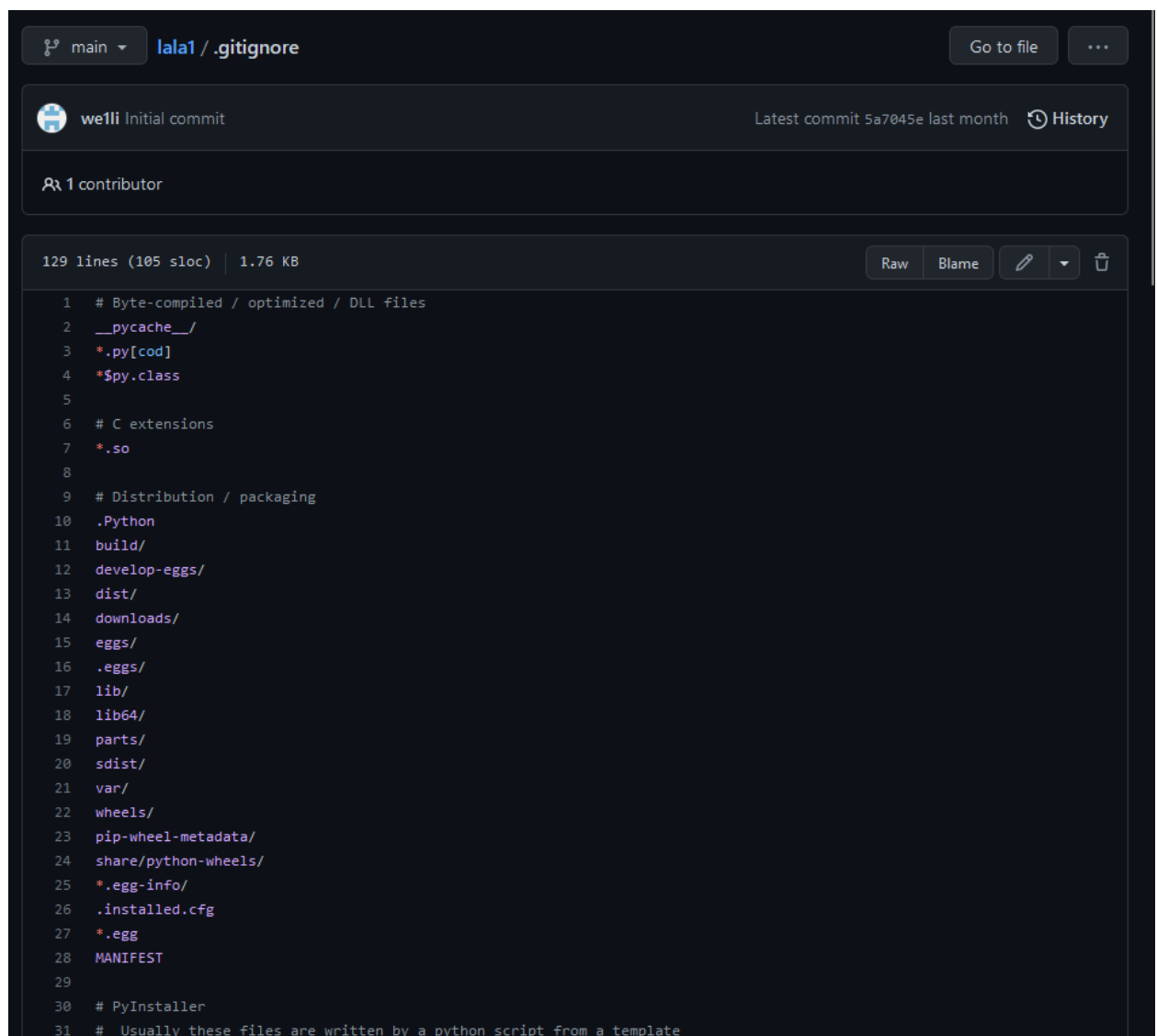
Комманда «git show v1.4».

Обмен тегами

```
C:\Users\user\Documents\GitHub\laba2>git push origin --tags
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 16 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 1.33 KiB | 453.00 KiB/s, done.
Total 10 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/we1li/laba2.git
 * [new tag]          v1.4 -> v1.4
```

Комманда «git push origin –tags».

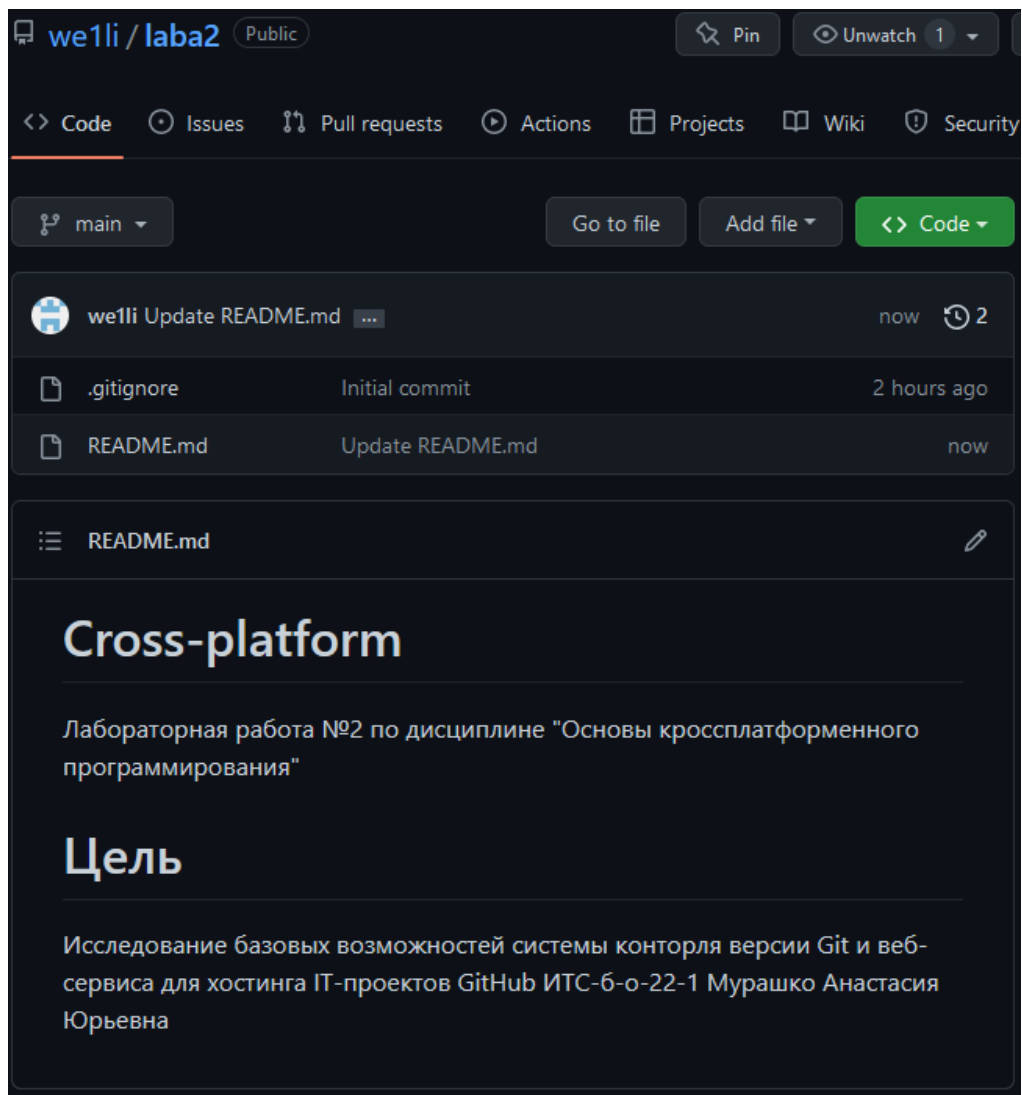
3. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.



The screenshot shows the GitHub web interface for a repository named 'laba1' by user 'we1li'. The file '.gitignore' is selected, showing its content. The file has 129 lines (105 sloc) and is 1.76 KB. The content lists various files and directories to be ignored, including compiled files, C extensions, distribution/packaging files, and build artifacts. The interface includes a 'Go to file' button, a 'History' link, and a '1 contributor' indicator.

```
1 # Byte-compiled / optimized / DLL files
2 __pycache__/
3 *.py[cod]
4 *$py.class
5
6 # C extensions
7 *.so
8
9 # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 pip-wheel-metadata/
24 share/python-wheels/
25 *.egg-info/
26 .installed.cfg
27 *.egg
28 MANIFEST
29
30 # PyInstaller
31 # Usually these files are written by a python script from a template
```

4. Добавьте в файл README.md информацию о группе и ФИО студента, выполняющего Лабораторную работу:



5. Напишите небольшую программу на выбранном Вами языке программирования. Фиксируйте изменения при написании программы в локальном репозитории. Должно быть сделано не менее 7 коммитов, отмеченных не менее 3 тэгами.

```
commit 30289144a64705d20a6bced24ddeebc66055a1f6 (HEAD -> main, origin/main, origin/HEAD)
Merge: 0063691 f3c5f2f
Author: Nastya <nastya.murashko1@yandex.ru>
Date: Tue Mar 14 23:33:42 2023 +0300

    Merge branch 'main' of https://github.com/we1li/laba2

commit 006369123dc8a74a18955a565747ec62ae0ff2a0
Author: Nastya <nastya.murashko1@yandex.ru>
Date: Tue Mar 14 23:29:43 2023 +0300

    add file programm

commit b72af1ffc4821652ed835d156462e63f5b2d78c5 (tag: v1.1, tag: v1.0)
Author: Nastya <nastya.murashko1@yandex.ru>
Date: Tue Mar 14 23:24:55 2023 +0300

    add file

commit f3c5f2f459f72c8d5b9ac476b37020f0be550f85
Author: we1li <125406661+we1li@users.noreply.github.com>
Date: Tue Mar 14 23:19:18 2023 +0300

    Update README.md

commit 822f4eb8f469f49517dda53d44057eb7ed75a11c (tag: v1.4)
Author: Nastya <nastya.murashko1@yandex.ru>
Date: Tue Mar 14 22:35:46 2023 +0300

    gitignore

commit 9acbfbba8853343f158a7bed209c80442c098cf6
Author: Nastya <nastya.murashko1@yandex.ru>
Date: Tue Mar 14 22:29:20 2023 +0300

    add 2 file

commit 36b34b358ace7188f06a67258bf8d239329c9fc1
Author: Nastya <nastya.murashko1@yandex.ru>
Date: Tue Mar 14 21:50:59 2023 +0300

    add file 26_1

commit cc99eb5862fc23292f2a2c59d8865e8ebc527ec0
Author: we1li <125406661+we1li@users.noreply.github.com>
Date: Tue Mar 14 21:05:15 2023 +0300

    Initial commit
(END)
commit f3c5f2f459f72c8d5b9ac476b37020f0be550f85
Author: we1li <125406661+we1li@users.noreply.github.com>
Date: Tue Mar 14 23:19:18 2023 +0300

    Update README.md

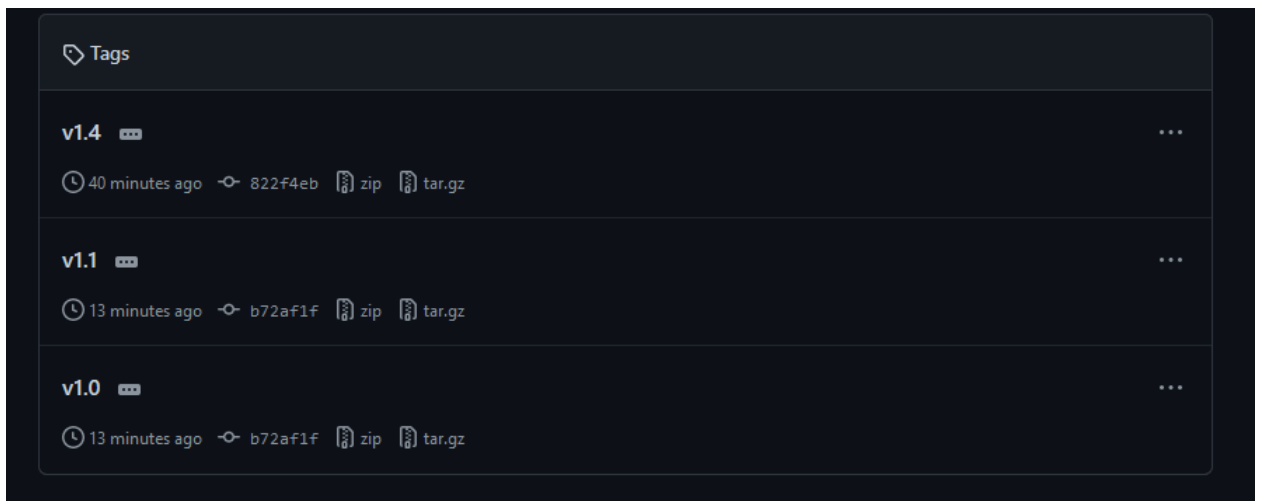
commit 822f4eb8f469f49517dda53d44057eb7ed75a11c (tag: v1.4)
Author: Nastya <nastya.murashko1@yandex.ru>
Date: Tue Mar 14 22:35:46 2023 +0300

    gitignore

commit 9acbfbba8853343f158a7bed209c80442c098cf6
Author: Nastya <nastya.murashko1@yandex.ru>
Date: Tue Mar 14 22:29:20 2023 +0300

    add 2 file

commit 36b34b358ace7188f06a67258bf8d239329c9fc1
Author: Nastya <nastya.murashko1@yandex.ru>
Date: Tue Mar 14 21:50:59 2023 +0300
```



Теги коммитов.

1. Что такое СКВ и каково ее назначение?

Система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

2. В чем недостатки локальных и централизованных СКВ?

Локальные: возможность потери данных вследствие возникновения физических поломок оборудования; отсутствие возможности совместной разработки. Централизованные: отсутствие доступа к данным при сбое работы сервера; довольно низкая скорость работы (из-за возникновения сетевых задержек).

3. К какой СКВ относится Git?

В Git каждая рабочая копия кода сама по себе является репозиторием.

4. В чем концептуальное отличие Git от других СКВ?

Бесплатный и open-source. Можно бесплатно скачать и вносить любые изменения в исходный код; Небольшой и быстрый. Выполняет все операции локально, что увеличивает его скорость. Кроме того, Git локально сохраняет весь репозиторий в небольшой файл без потери качества данных; Простое ветвление. В других системах контроля версий создание веток— утомительная и трудоёмкая задача, так как весь код копируется в новую ветку. В Git управление ветками реализовано гораздо проще и эффективнее.

5. Как обеспечивается целостность хранимых данных в Git?

Git обеспечивает целостность хранимых данных, используя контрольные суммы в качестве идентификаторов.

6. В каких состояниях могут находиться файлы в Git? Как связаны эти состояния?

Отслеживаемые файлы могут находиться в 3 состояниях: Не изменено (Unmodified), изменено (Modified), подготовленное (Staged).

7. Что такое профиль пользователя в GitHub?

У каждого пользователя есть публичный профиль, который помогает в поиске работы. Чтобы показать свой опыт потенциальному работодателю, нужно оставить в резюме ссылку на профиль. Когда рекрутер или другой специалист перейдут по ней, то увидят информацию о вас.

8. Какие бывают репозитории в GitHub?

Репозиторий Git бывает локальный и удалённый.

9. Укажите основные этапы модели работы с GitHub.

Установка Git; добавление имени, фамилии и адреса электронной почты; ввод определенных команд для Git; загрузка изменений в состояние (staged); добавление коммита; отправка в репозиторий на сервис GitHub.

10. Как осуществляется первоначальная настройка Git после установки?

Добавление имени, фамилии и адреса электронной почты: `git config --global user.name` – указывает ваше имя, фамилию. `git config --global user.email` – указывает вашу электронную почту. `git init` – создает новый репозиторий Git.

11. Опишите этапы создания репозитория в GitHub.

Ввод имени для репозитория, добавление описания проекта (выборочно), выбор приватности данного репозитория, добавление дополнительных файлов, как README.md и .gitignore.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Academic Free License v3.0; Boost Software License 1.0; Creative Commons license family; Eclipse Public License 1.0; ISC; MIT и многие другие.

13.Как осуществляется клонирование репозитория GitHub?

Зачем нужно клонировать репозиторий? С помощью команд `git clone/git push`.

Чтобы упростить устранение конфликтов слияния, добавление или удаление файлов и отправку больших фиксаций.

14.Как проверить состояние локального репозитория Git? Используйте команду `git status` , чтобы проверить текущее состояние репозитория