

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Основы программной инженерии**

**Отчет по лабораторной работе №2**

Исследование  
основных возможностей Git и GitHub

Выполнил студент группы

ПИЖ-б-о-21-1

Гасанов Г.М «    » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена «    » \_\_\_\_\_ 20\_\_ г.

Проверил доцент

Кафедры инфокоммуникаций, старший  
преподаватель

Воронкин Р.А.

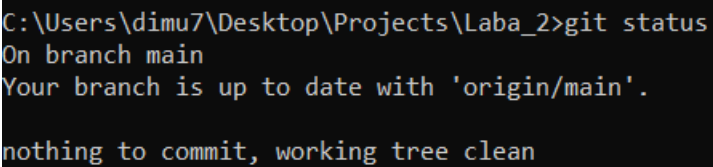
\_\_\_\_\_  
(подпись)

Ставрополь 2022

Тема: исследование возможностей Git для работы с локальными репозиториями.

Цель работы: исследовать базовые возможности системы контроля версий Git для работы с локальными репозиториями.

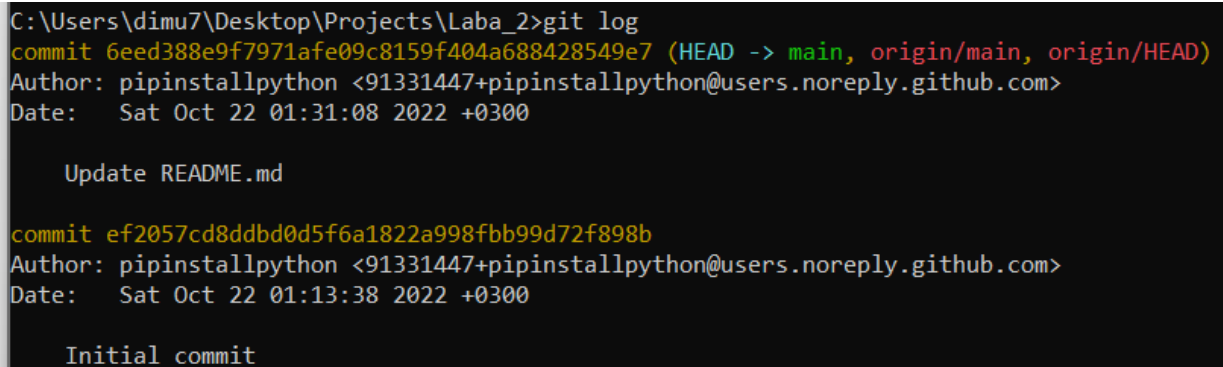
Выполнение работы.



```
C:\Users\dimu7\Desktop\Projects\Laba_2>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Рисунок 1. Клонирование репозитория



```
C:\Users\dimu7\Desktop\Projects\Laba_2>git log
commit 6eed388e9f7971afe09c8159f404a688428549e7 (HEAD -> main, origin/main, origin/HEAD)
Author: pipinstallpython <91331447+pipinstallpython@users.noreply.github.com>
Date: Sat Oct 22 01:31:08 2022 +0300

    Update README.md

commit ef2057cd8ddb0d5f6a1822a998fbb99d72f898b
Author: pipinstallpython <91331447+pipinstallpython@users.noreply.github.com>
Date: Sat Oct 22 01:13:38 2022 +0300

    Initial commit
```

Рисунок 2. Коммиты

```

C:\Users\dimu7\Desktop\Projects\Laba_2>git log -p -2
commit 6eed388e9f7971afe09c8159f404a688428549e7 (HEAD -> main, origin/main, origin/HEAD)
Author: pipinstallpython <91331447+pipinstallpython@users.noreply.github.com>
Date: Sat Oct 22 01:31:08 2022 +0300

    Update README.md

diff --git a/README.md b/README.md
index f8c9b67..1bd7b7f 100644
--- a/README.md
+++ b/README.md
@@ -1,2 @@
-# Laba_2
\ No newline at end of file
+# Laba_2
+Гасанов Гамид ПИЖ-6-о-21-1

commit ef2057cd8ddb0d5f6a1822a998fbb99d72f898b
Author: pipinstallpython <91331447+pipinstallpython@users.noreply.github.com>
Date: Sat Oct 22 01:13:38 2022 +0300

    Initial commit

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

```

Рисунок 3. Вывод коммитов с определённым условием

```

C:\Users\dimu7\Desktop\Projects\Laba_2>git log --stat
commit 6eed388e9f7971afe09c8159f404a688428549e7 (HEAD -> main, origin/main, origin/HEAD)
Author: pipinstallpython <91331447+pipinstallpython@users.noreply.github.com>
Date: Sat Oct 22 01:31:08 2022 +0300

    Update README.md

README.md | 3 ++-
1 file changed, 2 insertions(+), 1 deletion(-)

commit ef2057cd8ddb0d5f6a1822a998fbb99d72f898b
Author: pipinstallpython <91331447+pipinstallpython@users.noreply.github.com>
Date: Sat Oct 22 01:13:38 2022 +0300

    Initial commit

.gitignore | 129 +++++
LICENSE | 21 +++++
README.md | 1 +
3 files changed, 151 insertions(+)

```

Рисунок 4. Сокращенная статистика

```
C:\Users\dimu7\Desktop\Projects\Laba_2>git log --pretty=oneline
6eed388e9f7971afe09c8159f404a688428549e7 (HEAD -> main, origin/main, origin/HEAD) Update README.md
ef2057cd8ddb0d5f6a1822a998fbb99d72f898b Initial commit
```




Рисунок 5. Коммиты



```
C:\Users\dimu7\Desktop\Projects\Laba_2>git log --pretty=format:"%h - %an, %ar : %s"
6eed388 - pipinstallpython, 32 minutes ago : Update README.md
ef2057c - pipinstallpython, 49 minutes ago : Initial commit
```

Рисунок 6. Коммиты с определённым форматом

```
C:\Users\dimu7\Desktop\Projects\Laba_2>git log --pretty=format:"%h %s" --graph
* 6eed388 Update README.md
* ef2057c Initial commit
```

Рисунок 7. Текущая ветка и история слияний

 main ▾  1 branch  0 tags Go to file Add file ▾ Code ▾

 pipinstallpython Update README.md 6eed388 33 minutes ago  2 commits




 .gitignore	Initial commit	1 hour ago
 LICENSE	Initial commit	1 hour ago
 README.md	Update README.md	33 minutes ago

Рисунок 8. Репозиторий

```
1 name = input("Введите свое имя: ")
2 print("Привет,", name)
3 age = int(input("Сколько вам лет? "))
4 print(f"Через 17 лет вам будет {age + 17} лет")
5 if age > 20:
6     print(f"20 лет назад вам было {age - 20} лет")
```

Рисунок 9. Код

```
C:\Users\dimu7\Desktop\Projects\Laba_2>git log --graph --pretty=oneline --abbrev-commit
* 0dc1d86 (HEAD -> main, origin/main, origin/HEAD) Небольшие изменения
* 2827e96 (tag: v1.6) Небольшие изменения
* 3ec5c35 (tag: v1.5) Небольшие изменения
* a59704a (tag: v1.4) Добавлены ввод и вывод имени, также созданы папки code и doc
* eaad84b Небольшие изменения
* 6eed388 Update README.md
* ef2057c Initial commit
```

Рисунок 10. Просмотрел историю хранилища

```
C:\Users\dimu7\Desktop\Projects\Laba_2>git show HEAD
commit 0dc1d860b8b338a78454063b5d82158a69e19eb7 (HEAD -> main, origin/main, origin/HEAD)
Author: Gamid <dim.u7kin@yandex.ru>
Date: Sat Oct 22 02:24:02 2022 +0300

    Небольшие изменения

diff --git a/code/main.py b/code/main.py
index 8c9e216..c2aebf1 100644
--- a/code/main.py
+++ b/code/main.py
@@ -1,4 +1,6 @@
name = input("Введите свое имя: ")
print("Привет,", name)
age = int(input("Сколько вам лет? "))
-print(f"Через 17 лет вам будет {age + 17} лет")
\ No newline at end of file
+print(f"Через 17 лет вам будет {age + 17} лет")
+if age > 20:
+    print(f"20 лет назад вам было {age - 20} лет")
\ No newline at end of file
```

Рисунок 11. Просмотрел коммиты с помощью git show

```

C:\Users\dimu7\Desktop\Projects\Laba_2>git show 2827e96
commit 2827e96e96ddd0de6e3de487e5aff80bc70fa3da (tag: v1.6)
Author: Gamid <dim.u7kin@yandex.ru>
Date: Sat Oct 22 02:20:33 2022 +0300

    Небольшие изменения

diff --git a/code/main.py b/code/main.py
index be20e91..8c9e216 100644
--- a/code/main.py
+++ b/code/main.py
@@ -1,3 +1,4 @@
     name = input("Введите свое имя: ")
     print("Привет,", name)
-age = int(input("Сколько вам лет? "))
\ No newline at end of file
+age = int(input("Сколько вам лет? "))
+print(f"Через 17 лет вам будет {age + 17} лет")
\ No newline at end of file

```

Рисунок 12. Команда git show

```

C:\Users\dimu7\Desktop\Projects\Laba_2>git add .

C:\Users\dimu7\Desktop\Projects\Laba_2>git commit -m "Удаление кода"
[main 867d07c] Удаление кода
2 files changed, 6 deletions(-)
delete mode 100644 code/main.py
create mode 100644 main.py

C:\Users\dimu7\Desktop\Projects\Laba_2>git reset --hard
HEAD is now at 867d07c Удаление кода

```

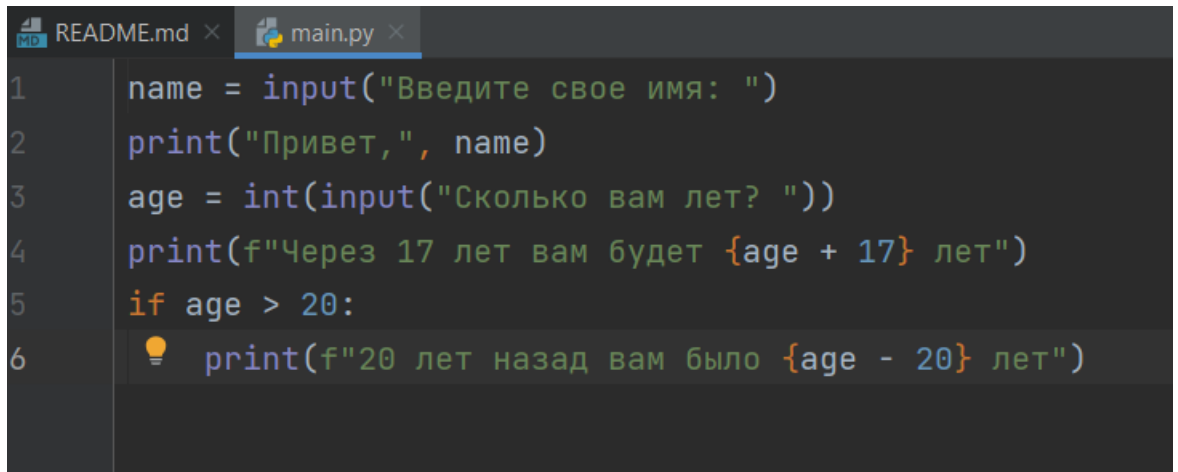
Рисунок 13. Удалённый код

```

C:\Users\dimu7\Desktop\Projects\Laba_2>git reset --hard HEAD~1
HEAD is now at 0dc1d86 Небольшие изменения

```

Рисунок 14. Откат версии



```
1 name = input("Введите свое имя: ")
2 print("Привет,", name)
3 age = int(input("Сколько вам лет? "))
4 print(f"Через 17 лет вам будет {age + 17} лет")
5 if age > 20:
6     print(f"20 лет назад вам было {age - 20} лет")
```

Рисунок 15. Восстановленный код

Вывод: команда `git -checkout <FileName>` удаляет изменения произошедшие с файлом в репозитории до коммита.

1. Как выполнить историю коммитов в Git? Какие существуют дополнительные опции для просмотра истории коммитов?

Наиболее простой и в то же время мощный инструмент для этого — команда `git log`. По умолчанию, без аргументов, `git log` выводит список коммитов созданных в данном репозитории в обратном хронологическом порядке. То есть самые последние коммиты показываются первыми. Одна из опций, когда вы хотите увидеть сокращенную статистику для каждого коммита, вы можете использовать опцию `—stat`. Вторая опция (одна 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 (пример команды `git log -p -2`). Третья действительно полезная опция это `--pretty`. Она 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 соответственно. Наиболее интересной опцией является `format`, которая позволяет указать формат для вывода информации. Особенно это может быть полезным, когда вы хотите сгенерировать вывод для автоматического анализа — так как вы указываете формат явно, он не будет изменен даже после обновления Git. Для опции `git log --pretty=format` существуют различного рода опции для изменения формата отображения.

## 2. Как ограничить вывод при просмотре истории коммитов?

Для ограничения может использоваться функция `git log`, где `n` — число записей. Также, существуют опции для ограничения вывода по времени, такие как `--since` и `--until`, они являются очень удобными. Например, следующая команда покажет список коммитов, сделанных за последние две недели: `git log --since=2.weeks`. Эта команда работает с большим количеством форматов — вы можете указать определенную дату вида `2008-01-15` или же относительную дату, например `2 years 1 day 3 minutes ago`. Также вы можете фильтровать список коммитов по заданным параметрам. Опция `--author` дает возможность фильтровать по автору коммита, а опция `--grep` (показывает только коммиты, сообщение которых содержит указанную строку) искать по ключевым словам в сообщении коммита.<sup>8</sup> Функция `-S` показывает только коммиты, в которых изменение в коде повлекло за собой добавление или удаление указанной строки.

## 3. Как внести изменения в уже сделанный коммит?

Внести изменения можно с помощью команды `git commit --amend`. Эта команда берёт индекс и применяет его к последнему



коммиту. Если после последнего коммита не было никаких проиндексированных изменений (например, вы запустили приведённую команду сразу после предыдущего коммита), то состояние проекта будет абсолютно таким же и всё, что мы изменим, это комментарий к коммиту. Для того, чтобы внести необходимые изменения - нам нужно проиндексировать их и выполнить команду `git commit --amend`. `git commit -m 'initial commit' git add forgotten_file git commit --amend` Эффект от выполнения этой команды такой, как будто мы не выполнили предыдущий коммит, а еще раз выполнили команду `git add` и выполнили коммит.

#### 4. Как отменить индексацию файла в Git?

Например, вы изменили два файла и хотите добавить их в разные коммиты, но случайно выполнили команду `git add *` и добавили в индекс оба. Как исключить из индекса один из них? Команда `git status` напомним вам: Прямо под текстом «Changes to be committed» говорится: используйте `git reset HEAD` для исключения из индекса.

#### 5. Как отменить изменения в файле?

С помощью команды `git checkout --`.

#### 6. Что такое удаленный репозиторий Git?

Удалённый репозиторий это своего рода наше облако, в которое мы сохраняем те или иные изменения в нашей программе/коде/файлах. 7.

Как выполнить просмотр удаленных репозиториях данного локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозиториях, необходимо запустить команду `git remote`. Также можно указать ключ `-v`, чтобы просмотреть адреса для чтения и записи, привязанные к репозиторию. Пример: `git remote -v`

8. Как добавить удаленный репозиторий для данного локального репозитория?

Для того, чтобы добавить удалённый репозиторий и присвоить ему имя (shortname), просто выполните команду `git remote add` .

9. Как выполнить отправку/получение изменений с удаленного репозитория?

Если необходимо получить изменения, которые есть у Пола, но нету у вас, вы можете выполнить команду `git fetch` . Важно отметить, что команда `git fetch` забирает данные в ваш локальный репозиторий, но не сливает их с какими-либо вашими наработками и не модифицирует то, над чем вы работаете в данный момент. Вам необходимо вручную слить эти данные с вашими, когда вы будете готовы. Если ветка настроена на отслеживание удалённой ветки, то вы можете использовать команду `git pull` чтобы автоматически получить изменения из удалённой ветки и слить их со своей текущей. Выполнение `git pull`, как правило, извлекает (fetch) данные с сервера, с которого вы изначально клонировали, и автоматически пытается слить (merge) их с кодом, над которым вы в данный момент работаете. Чтобы отправить изменения на удалённый репозиторий необходимо отправить их в удалённый репозиторий. Команда для этого действия простая: `git push` .

10. Как выполнить просмотр удаленного репозитория?

Для просмотра удалённого репозитория, можно использовать команду `git remote show` .

11. Каково назначение тэгов Git?

Теги - это ссылки указывающие на определённые версии кода/написанной программы. Они удобны чтобы в случае чего вернуться к нужному моменту. Также при помощи тегов можно помечать важные моменты.

12. Как осуществляется работа с тэгами Git?

Просмотреть наличие тегов можно с помощью команды: `git tag`. А назначить (указать, добавить тег) можно с помощью команды `git tag -a v1.4(версия изначальная) -m "Название"`. С помощью команды `git show` вы можете посмотреть данные тега вместе с коммитом: `git show v1.4`. Отправка тегов, по умолчанию, команда `git push` не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на удалённый сервер. Процесс аналогичен отправке веток — достаточно выполнить команду `git push origin`. Для отправки всех тегов можно использовать команду `git push origin tags`. Для удаления тега в локальном репозитории достаточно выполнить команду `git tag -d`. Например, удалить созданный ранее легковесный тег можно следующим образом: `git tag -d v1.4-lw`. Для удаления тега из внешнего репозитория используется команда `git push origin --delete`. Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать `git checkout` для тега пример: `git checkout -b version2 v2.0.0`.

13. Самостоятельно изучите назначение флага `--prune` в командах `git fetch` и `git push`. Каково назначение этого флага?

`Git fetch --prune` команда получения всех изменений с репозитория GitHub. В команде `git push --prune` удаляет удаленные ветки, у которых нет локального аналога. Вывод: исследовал базовые возможности системы контроля версий `git` для работы с локальными репозиториями. Также, благодаря созданию тегов и пункту 7 лабораторной работы после изменения файлов освоил возможность отката к заданной версии.

Вывод: исследовал базовые возможности системы контроля версий `git` для работы с локальными репозиториями. Освоил возможность отката измененных файлов.