

Федеральное государственное автономное образовательное учреждение
высшего образования «Северо-Кавказский федеральный университет»
Институт цифрового развития

Отчет по дисциплине основы программной инженерии

Выполнила:
Первых Дарья
Александровна,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Роман Александрович

ВЫПОЛНЕНИЕ

1. Работа с консолью Git

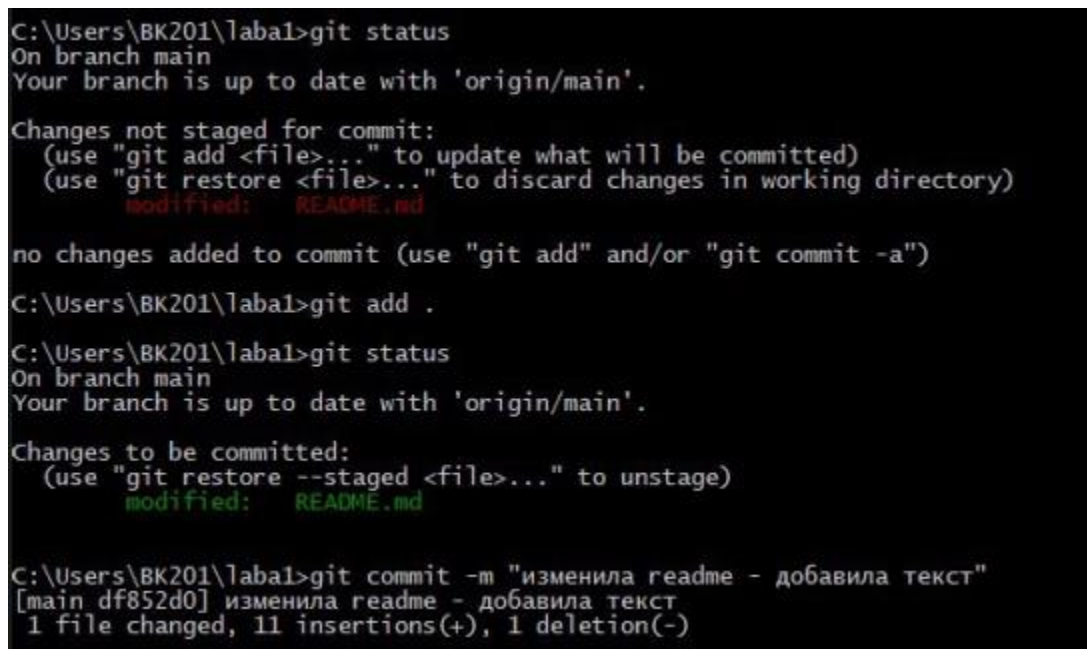
1) Клонирование репозитория на компьютер показано на рисунке 1.



```
Git CMD
C:\Users\BK201>git clone https://github.com/PervykhDarya/labal.git
Cloning into 'labal'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 1 – Результат клонирования репозитория.

2) Внесение изменений в файл README.md в локальном репозитории и отправка изменений на сервер показана на рисунках 2,3,4.



```
C:\Users\BK201\labal>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

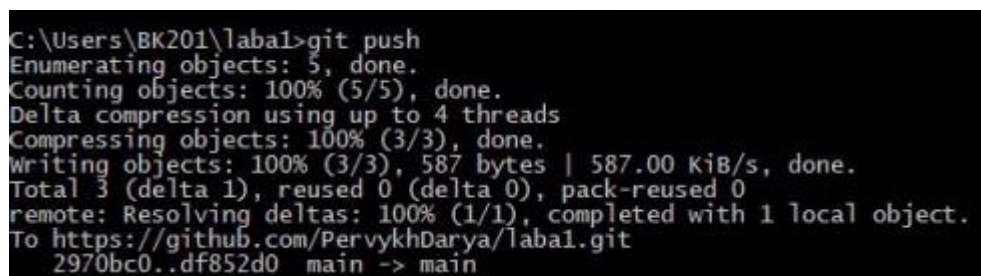
C:\Users\BK201\labal>git add .

C:\Users\BK201\labal>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

C:\Users\BK201\labal>git commit -m "изменила readme - добавила текст"
[main df852d0] изменила readme - добавила текст
1 file changed, 11 insertions(+), 1 deletion(-)
```

Рисунок 2 – Коммит изменений в файле.



```
C:\Users\BK201\labal>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 587 bytes | 587.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/PervykhDarya/labal.git
   2970bc0..df852d0  main -> main
```

Рисунок 3 - Процесс загрузки файлов на удаленный сервер.

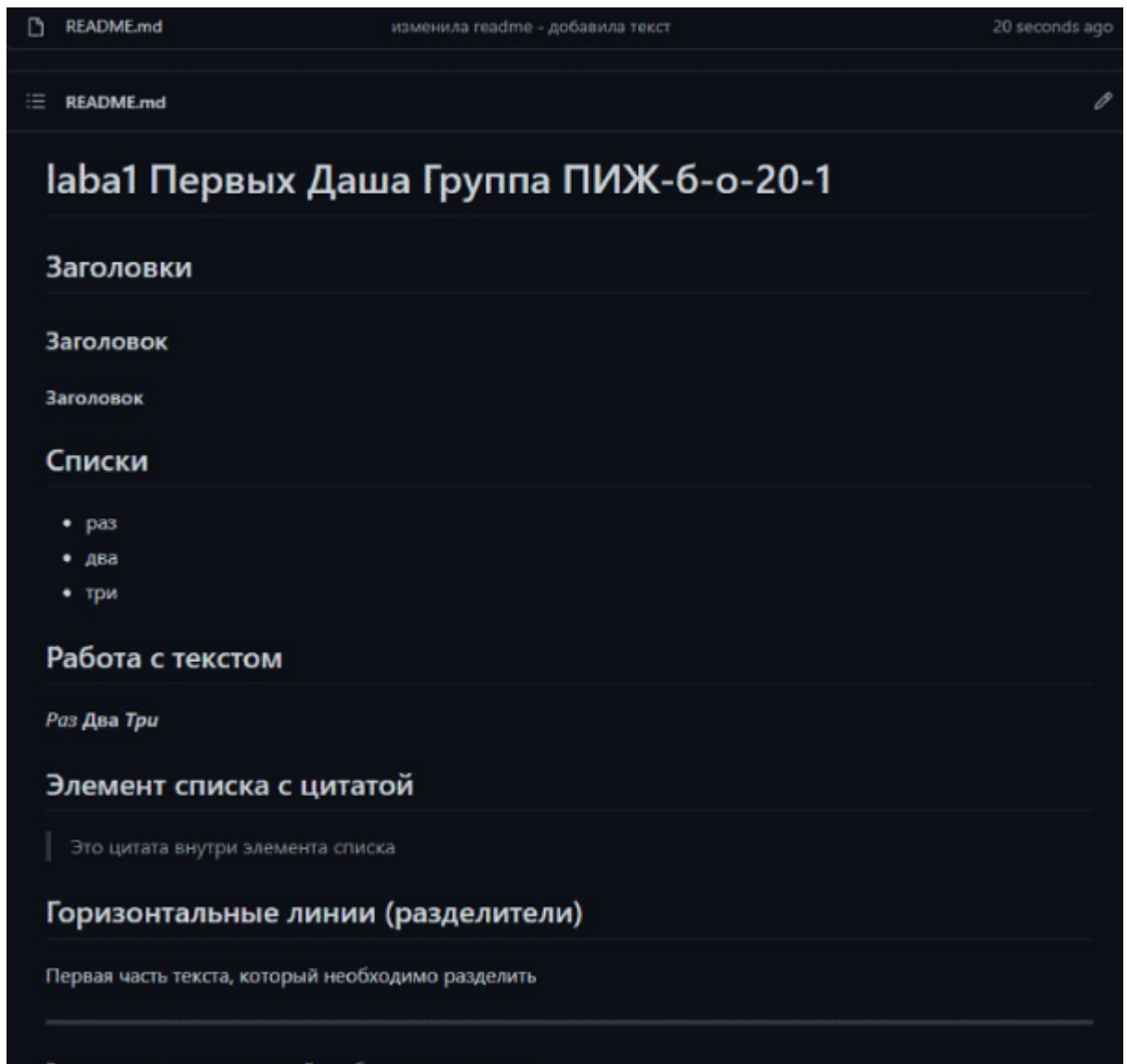


Рисунок 4 – Обновление файла в удалённом репозитории.

3) Добавление текстового файла «lyrics» и отправка на сервер показана на рисунках 5,6.

```
C:\Users\BK201\laba1>git add .
C:\Users\BK201\laba1>git commit -m "добавила текст в lyrics"
[main 0e8db6b] добавила текст в lyrics
1 file changed, 11 insertions(+)
C:\Users\BK201\laba1>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 520 bytes | 520.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/PervykhDarya/laba1.git
   3dbebea..0e8db6b  main -> main
```

Рисунок 5 – Добавление файла и отправка на удалённый сервер.

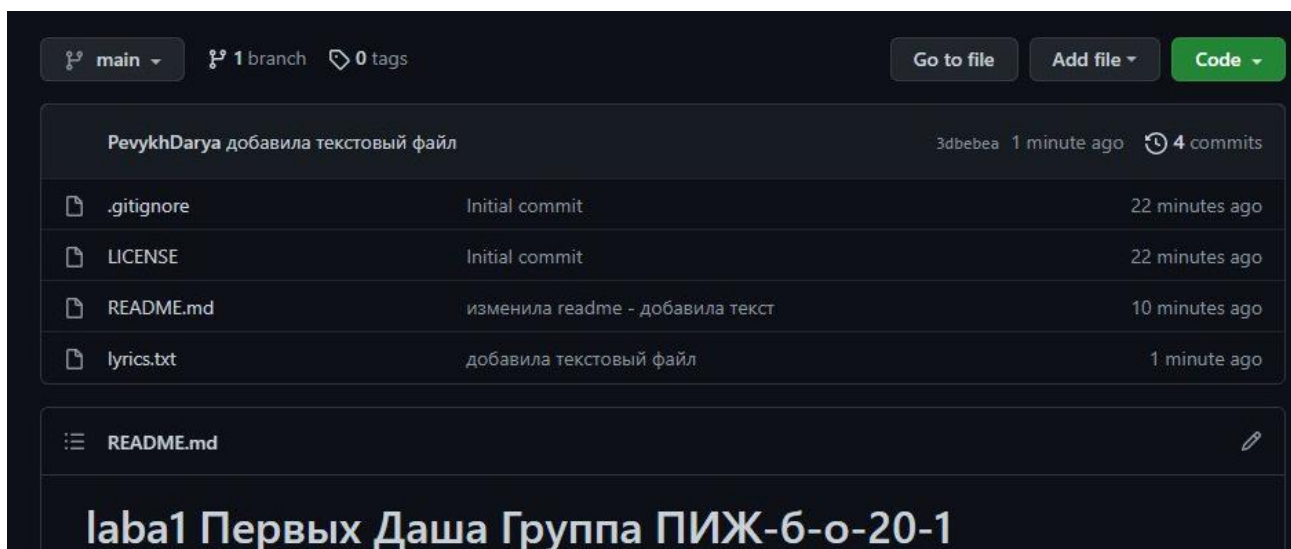


Рисунок 6 – Обновление списка файлов в удалённом репозитории.

4) Изменение в текстовом файле и отправка изменений на сервер показана на рисунках 7,8.

```
C:\Users\BK201\laba1>git add .
C:\Users\BK201\laba1>git commit -m "добавила исполнителя и название"
[main c0e6997] добавила исполнителя и название
1 file changed, 2 insertions(+)
C:\Users\BK201\laba1>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 373 bytes | 373.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/PevykhDarya/laba1.git
0e8db6b..c0e6997 main -> main
```

Рисунок 7 – Изменение и отправка на сервер.

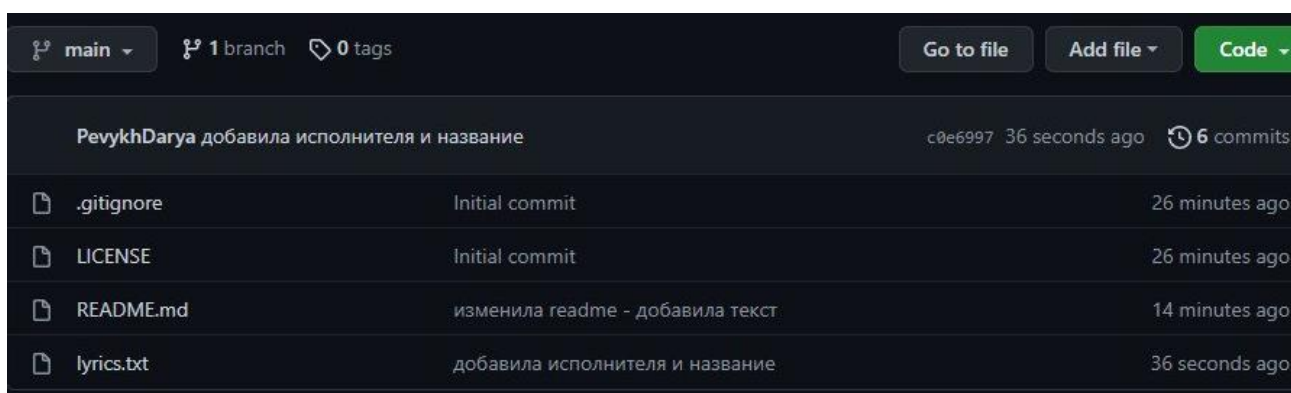
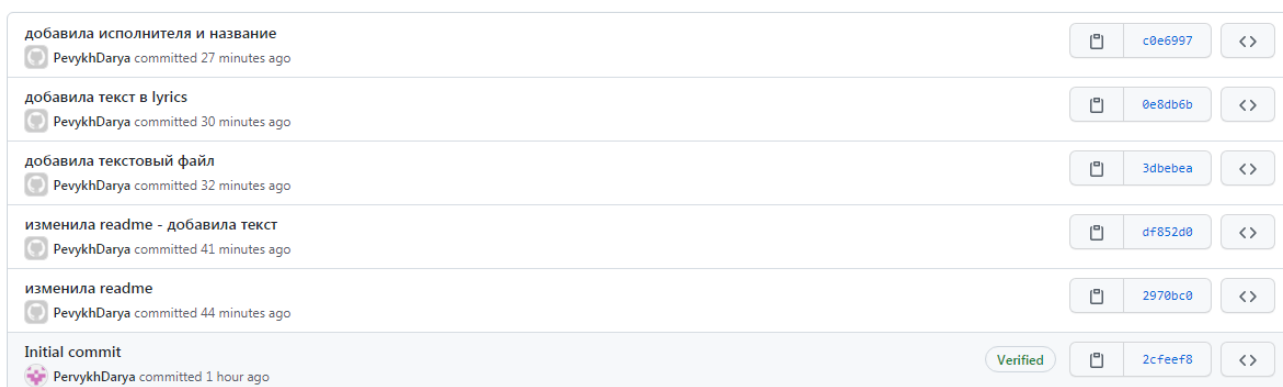


Рисунок 8 – Обновление файлов в удалённом репозитории.

5) Проверка системы контроля версий после внесения 6-ти коммитов показано на рисунке 9.



добавила исполнителя и название PevykhDarya committed 27 minutes ago	c0e6997	<>
добавила текст в lyrics PevykhDarya committed 30 minutes ago	0e8db6b	<>
добавила текстовый файл PevykhDarya committed 32 minutes ago	3dbebea	<>
изменила readme - добавила текст PevykhDarya committed 41 minutes ago	d7852d0	<>
изменила readme PevykhDarya committed 44 minutes ago	2970bc0	<>
Initial commit PevykhDarya committed 1 hour ago	Verified 2cfeef8	<>

Рисунок 9 – Список всех загруженных версий исходного файла в удаленном репозитории.

2. Ответы на вопросы

1. Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ) — это 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?

Проблемой локальной СКВ является основное свойство — локальность. Она совершенно не предназначена для коллективного использования.

Самый очевидный минус централизованной СКВ - это единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений.

3. К какой СКВ относится Git? Распределённой

4. В чем концептуальное отличие Git от других СКВ?

Основное отличие Git от любой другой СКВ — это подход к работе со своими данными. Концептуально, большинство других систем хранят информацию в виде списка изменений в файлах. Эти системы представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени (обычно это называют контролем версий, основанным на различиях). Git не хранит и не обрабатывает данные таким способом. Вместо этого, подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создаёт ссылку на предыдущую версию идентичного файла, который уже сохранён. Git представляет свои данные как, скажем, поток снимков.

5. Как обеспечивается целостность хранимых данных в Git?

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом.

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

У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged). Зафиксированный значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

7. Что такое профиль пользователя в GitHub?

Профиль - это публичная страница на GitHub, как и в социальных сетях. Когда вы ищете работу в качестве программиста, работодатели могут посмотреть ваш профиль GitHub и принять его во внимание, когда будут решать, брать вас на работу или нет.

8. Какие бывают репозитории в GitHub?

Общедоступные и приватные.

9. Укажите основные этапы модели работы с GitHub.

- 1) Создание аккаунта.
- 2) Создание репозитория.
- 3) Клонирование репозитория.
- 4) Локальное изменение содержимого.
- 5) Отправка изменений в удаленный репозиторий с помощью Git.

Стандартный подход к работе с проектом состоит в том, чтобы иметь локальную копию репозитория и фиксировать изменения в этой копии, а не в удаленном репозитории, размещенном на GitHub. Этот локальный репозиторий имеет полную историю версий проекта, которая может быть полезна при разработке без подключения к интернету.

10. Как осуществляется первоначальная настройка Git после установки?

Чтобы убедиться, что Git установлен, нужно написать в консоли команду «git version». Затем нужно добавить в Git имя пользователя и почту с GitHub.

Перед первой отправкой на сервер необходимо передать локальную ветку с помощью следующей команды: `git push -- set-upstream origin edit-readme`.

11. Опишите этапы создания репозитория в GitHub.

В правом верхнем углу нажать на кнопку создания нового репозитория. Затем указать его название, описание и выбрать необходимые предустановки, такие как: вид репозитория (открытый/закрытый), создание файла `.gitignore` и выбор лицензии.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Apache License 2.0
GNU General Public License v3.0
MIT License
BSD 2-Clause "Simplified" License
BSD 3-Clause "New" or "Revised" License
Boost Software License 1.0
Creative Commons Zero v1.0 Universal
Eclipse Public License 2.0
GNU Affero General Public License v3.0
GNU General Public License v2.0
GNU Lesser General Public License v2.1
Mozilla Public License 2.0

13. Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий?

После создания репозитория его необходимо клонировать на компьютер. Для этого на странице репозитория необходимо найти кнопку `Clone` или `Code` и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования. Клонирование выполняется для внесения в репозиторий локальных изменений и, наконец, запроса на обновление файлов в удаленном репозитории.

14. Как проверить состояние локального репозитория Git?

С помощью команды `«git status»`.

15. Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/измененного файла под версионный

контроль с помощью команды `git add`; фиксации(коммита) изменений с помощью команды `git commit` и отправки изменений на сервер с помощью команды `git push`?

При добавлении/изменении файлов они помечаются как «modified». При добавлении под версионный контроль файл помечается как «new file». При фиксации изменений статус меняется на «... ahead on 1 commit», после отправки статус выдает: «your branch is up to date».

16. У вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы можете осуществлять работу над некоторым проектом с использованием этого репозитория. Опишите последовательность команд, с помощью которых оба локальных репозитория, связанных с репозиторием GitHub будут находиться в синхронизированном состоянии. Примечание: описание необходимо начать с команды `git clone`.

Необходимо клонировать исходный репозиторий на каждый компьютер с помощью команды `git clone`.

На каждом компьютере после внесения локальных изменений нужно добавлять их под версионный контроль с помощью `git add`, делать коммит изменений (`git commit -m`), отправлять изменения на сервер (`git push`).

Для получения новых версий файлов из репозитория необходимо использовать команду «`git pull`».

17. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.

GitLab, SourceForge, BitBucket, Launchpad, FogCreek/DevHub, BeanStalk, GitKraken.

GitHub имеет удобный интерфейс, прост в использовании, но в отличие от sourceforge работает только с Git.

18. Интерфейс командной строки является не единственным и далеко не самым удобным способом работы с Git. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git? Приведите как реализуются описанные в лабораторной работе операции Git спомощью одного из таких программных средств.

Fork, Tower, Sourcetree, SmartGit, GitKraken и т.д. В таких программах действия, выполняемые с помощью консольных команд, представлены в пользовательском интерфейсе, что значительно упрощает знакомство с Git идальнейшее использовании.