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 №4
дисциплины «Основы программной инженерии»

Выполнила:
Рядская Мария александровна
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р. А., доцент кафедры
инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Ход работы

1. Я изучила теоретический материал работы

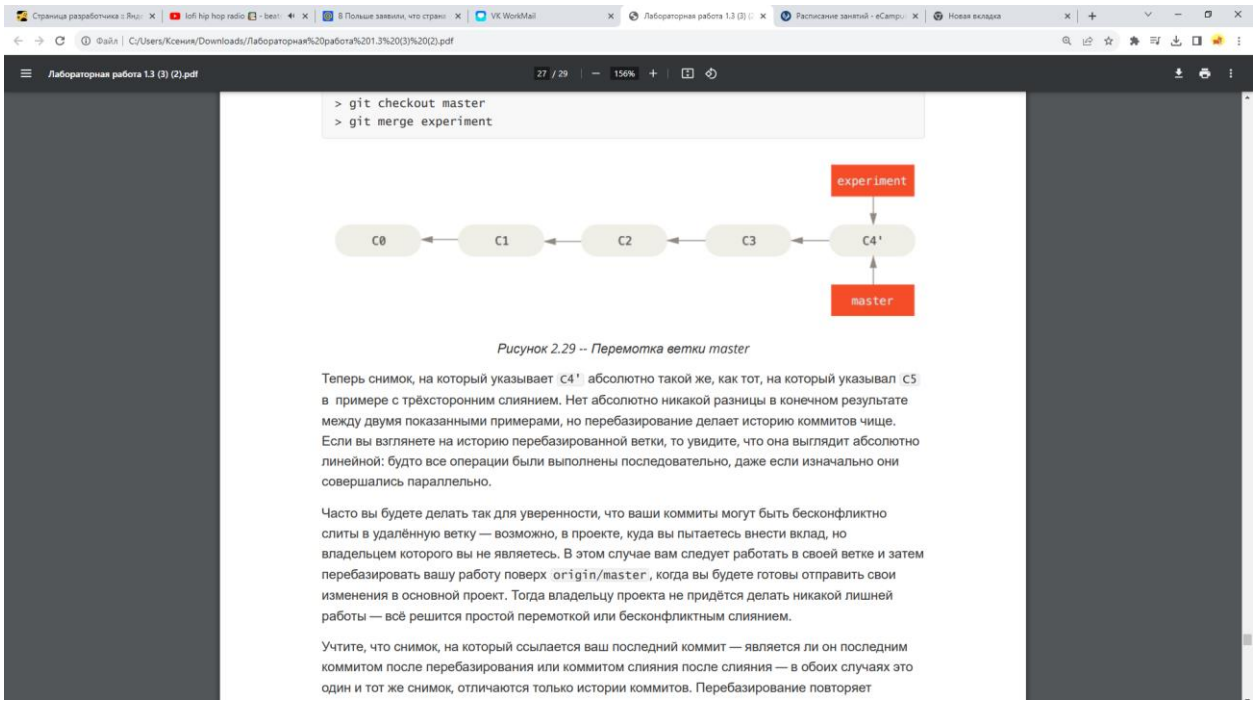

Рисунок 1.1 – Изучение материала для лабораторной работы

2. Создала общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный мною язык программирования

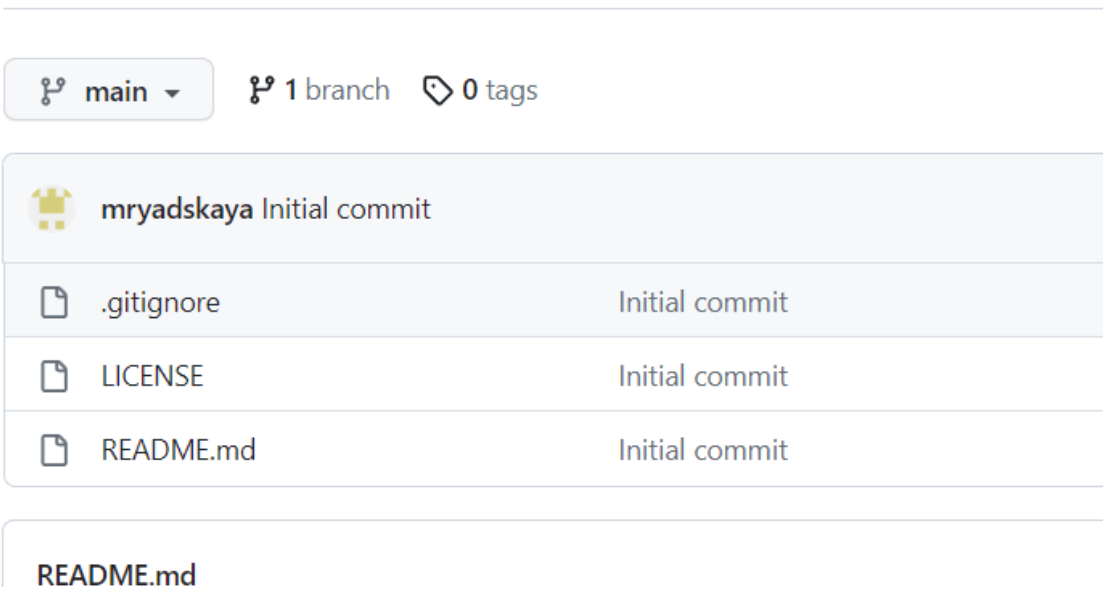

Рисунок 2 – Готовый репозиторий

3. Создаю три файла: 1.txt, 2.txt, 3.txt

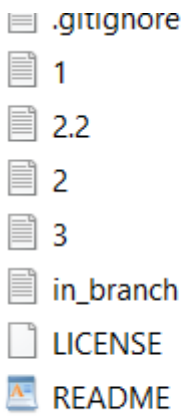

Рисунок 3.1 – Папка репозитория

4. Проиндексировала первый файл и сделала коммит с комментарием "add 1.txt file"

```
C:\git1>git clonehttps://github.com/mryadskaya/lab3.git
git: 'clonehttps://github.com/mryadskaya/lab3.git' is not a git command. See 'git --help'.

C:\git1>git clone https://github.com/mryadskaya/lab3.git
Cloning into 'lab3'...
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 4.1 – Индексирование и коммит файла 1.txt

5. Проиндексировала второй и третий файлы.

```
C:\TestGit\Iwasabandoned>git add 2.txt 3.txt

C:\TestGit\Iwasabandoned>
```

Рисунок 5.1 – Индексация файлов 2.txtи 3.txt

6. Перезаписала уже сделанный коммит с новым комментарием "add 2.txt and 3.txt."

```
C:\TestGit\Iwasabandoned>git commit -m"add 2.txt and 3.txt."
[main b00a1d7] add 2.txt and 3.txt.
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 3.txt
```

Рисунок 6.1 – Коммит файлов 2.txtи 3.txt

7. Создала новуюветку my_first_branch

```
fatal: unrecognized argument: --decoratee

C:\git1\lab3>git log --oneline --decorate
53626a3 (HEAD -> main, my_1) 2.py and 3.py
993b2c9 add 1.py file
f810879 (origin/main, origin/HEAD) Initial commit

C:\git1\lab3>git checkout my_1
Switched to branch 'my_1'

C:\git1\lab3>git add .

C:\git1\lab3>git commit -m"commit on branch my_1
[my_1 e99eb08] commit on branch my_1
 1 file changed, 1 insertion(+)
 create mode 100644 in_1.py
```

Рисунок 7.1 – Создание ветки

8. Перехожу на ветку и создаю новый файл in_branch.txt, коммичу изменения

```
C:\git1\lab3>git checkout my_1
Switched to branch 'my_1'

C:\git1\lab3>git add
```

Рисунок 8.1 – Переход на ветку my_first_branch

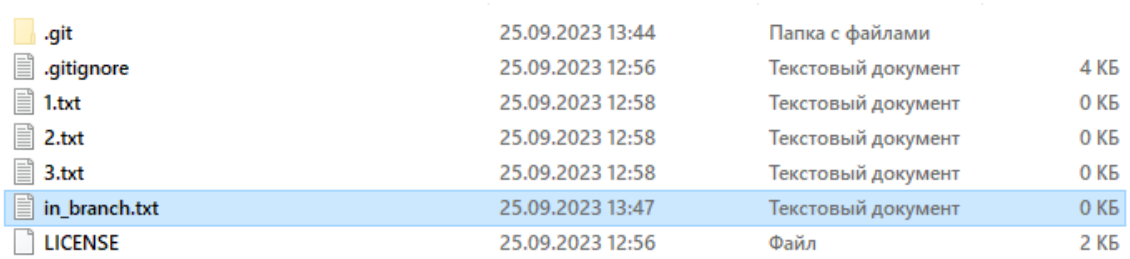

Рисунок 8.2 – Создание файла in_branch.txt

```
C:\git1\lab3>git add .

C:\git1\lab3>git commit -m"commit on branch my_1
[my_1 e99eb08] commit on branch my_1
 1 file changed, 1 insertion(+)
 create mode 100644 in_1.py

C:\git1\lab3>git checkout main
```

Рисунок 8.3 – Коммит изменений

9. Возвращаюсь на ветку main

```
C:\git1\lab3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

C:\git1\lab3>git checkout -b new_m
Switched to a new branch 'new_m'
```

Рисунок 9.1 – Переход на главную ветку

10. Создаю и сразу перехожу на ветку new_branch

```
C:\git1\lab3>git checkout -b new_m
Switched to a new branch 'new_m'
```

Рисунок 10.1 – Создание и переход на новую ветку

11. Делаю изменения в файле 1.txt, добавляю строчку “new row in the 1.txt file”,коммичу изменения

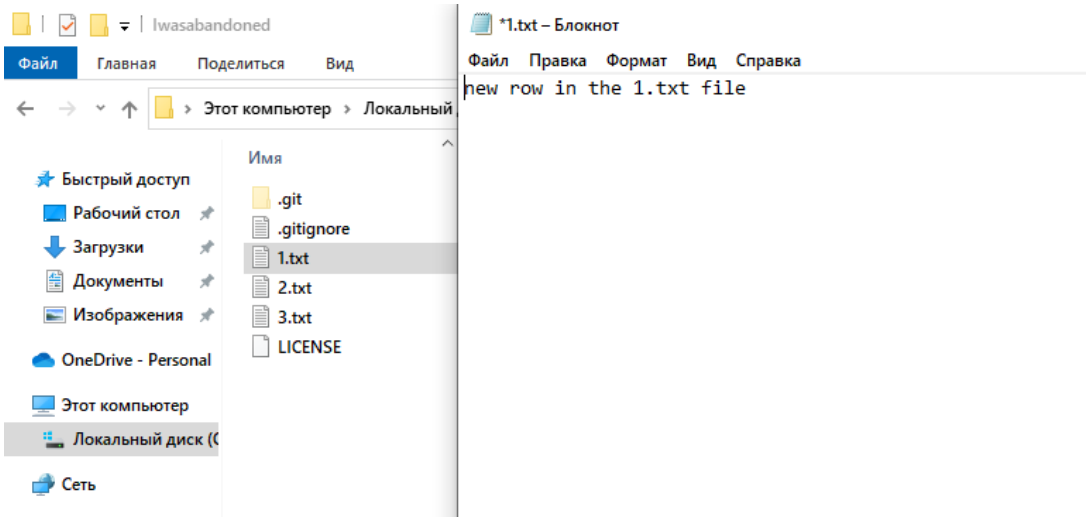

Рисунок 11.1 – Изменение файла 1.txtдобавлением строки

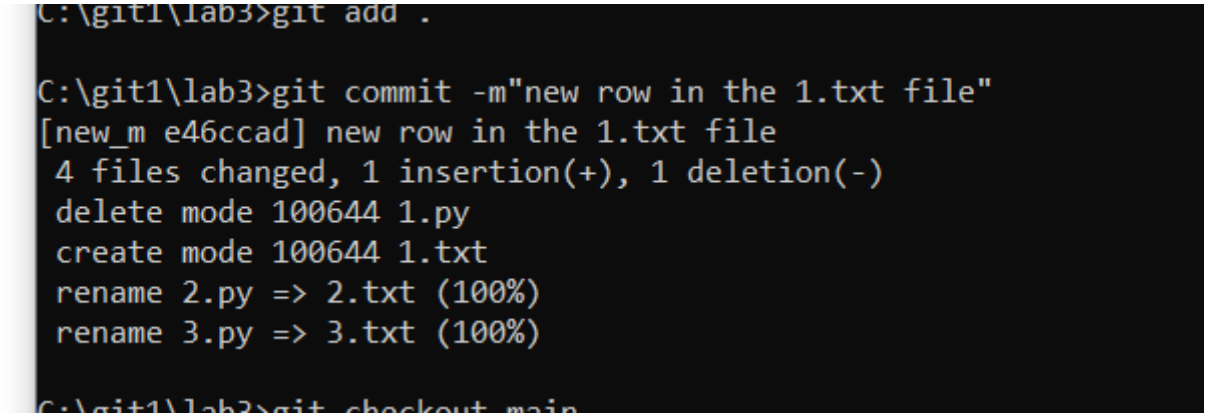

Рисунок 11.2 – Коммит изменений

12. Перейтинаветку mainислитьветки master и my_first_branch, послечегослитьветкиmainи new_branch

```
C:\git1\lab3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

C:\git1\lab3>git merge my_1
Updating 53626a3..e99eb08
Fast-forward
 in_1.py | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 in_1.py
```

Рисунок 12.1 – Переход на ветку mainи слияние с веткой my_first_branch

```
C:\git1\lab3>git merge my_1
Updating 53626a3..e99eb08
Fast-forward
 in_1.py | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 in_1.py
```

Рисунок 12.2 – Слияние с веткой new_branch

13. Удаляюветки my_first_branch и new_branch

```
C:\git1\lab3>git branch -d my_1
Deleted branch my_1 (was e99eb08).
```

Рисунок 13.1 – Удаление ненужных веток

14. Создаю ветки branch_1 иbranch_2

```
C:\git1\lab3>git branch branch_1

C:\git1\lab3>git branch branch_2

C:\git1\lab3>git log --oneline --decorate
a9e0d9e (HEAD -> main, branch_2, branch_1) Merge branch 'new_m'
e46ccad (new_m) new row in the 1.txt file
e99eb08 commit on branch my_1
53626a3 2.py and 3.py
993b2c9 add 1.py file
f810879 (origin/main, origin/HEAD) Initial commit
```

Рисунок 14.1 – Создание двух новых веток

15. Перехожу на ветку branch_1 и изменяю файл 1.txt, удаляю все содержимое и добавляю текст “fix in the 1.txt”, изменяю файл 3.txt, удаляю все содержимое и добавляю текст “fix in the 3.txt”, коммичу изменения.

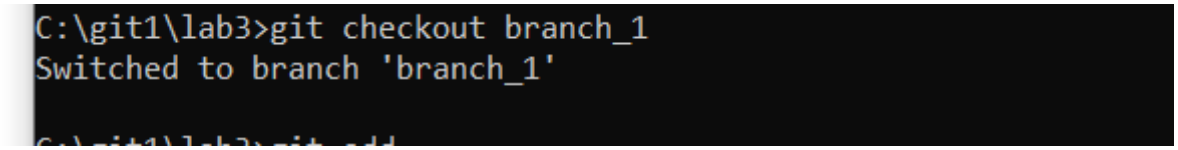

Рисунок 11.2 – Перехожу на ветку branch_1

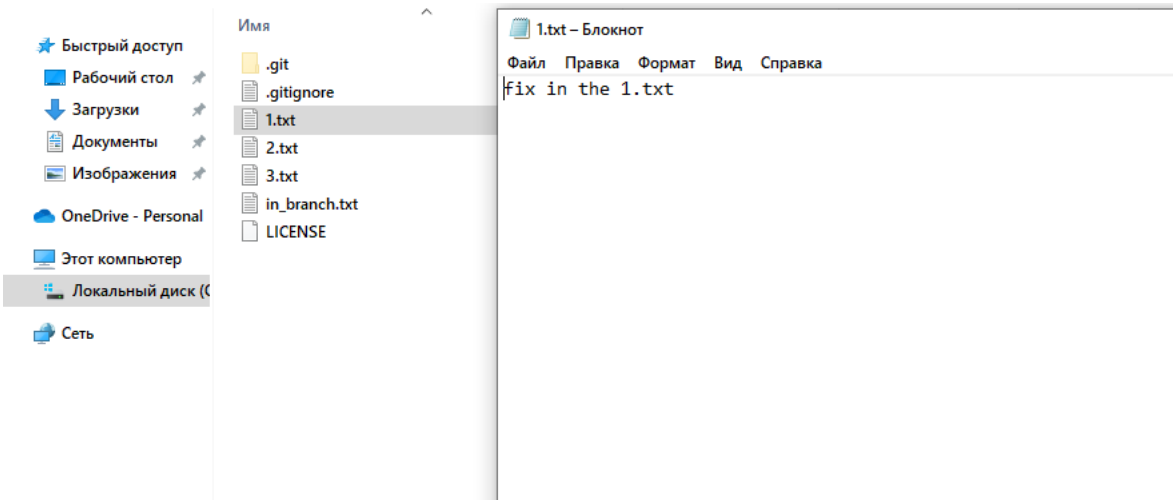

Рисунок 15.1 – Изменяю файл 1.txt, удаляя все содержимое и добавляя текст

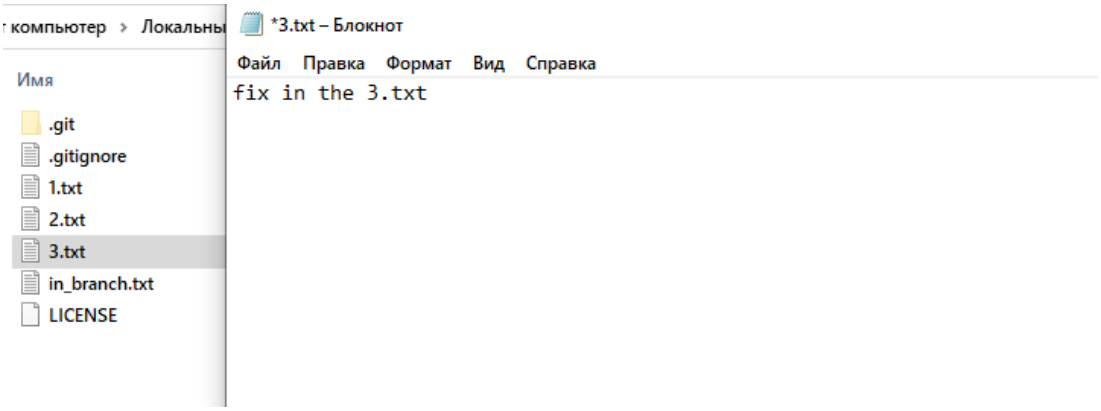

Рисунок 15.2 – Изменяю файл 3.txt, удаляя все содержимое и добавляя текст

```
C:\git1\lab3>git add .

C:\git1\lab3>git commit -m"change files 1.txt and 3.txt"
[branch_1 a344787] change files 1.txt and 3.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename in_1.py => in_branch.txt (100%)
```

Рисунок 15.3 – Коммит изменений

16. Перехожу на ветку branch_2 и также изменяю файл 1.txt, удаляю все содержимое и добавляю текст “My fix in the 1.txt”, изменяю файл 3.txt, удаляю все содержимое и добавляю текст “My fix in the 3.txt”, коммичу изменения.

```
C:\git1\lab3>git checkout branch_2
Switched to branch 'branch_2'
```

Рисунок 16.1 – Переход на другую ветку

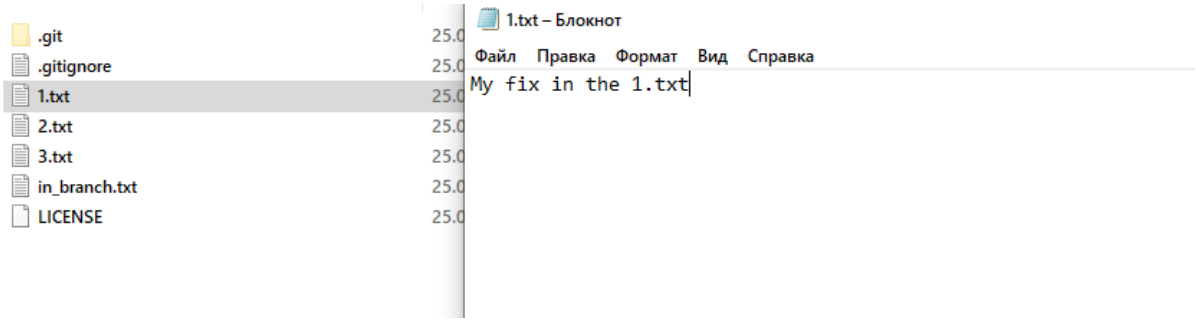

Рисунок 16.2 – Изменяю файл 1.txt, удаляя все содержимое и добавляя текст

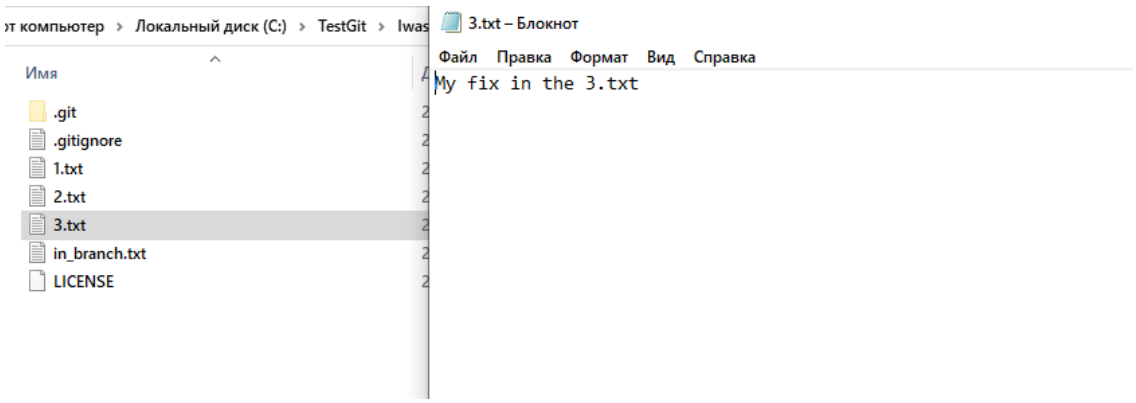

Рисунок 16.3 – Изменяю файл 3.txt, удаляя все содержимое и добавляя текст

```
C:\git1\lab3>git add .

C:\git1\lab3>git commit -m"change files on branch 2 1.txt and 3.txt"
[branch_2 81b5298] change files on branch 2 1.txt and 3.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 git
```

Рисунок 16.4 – Коммит изменений

17. Сливаю изменения ветки branch_2 в ветку branch_1

```
 create mode 100644 git

C:\git1\lab3>git checkout branch_1
Switched to branch 'branch_1'

C:\git1\lab3>git merge branch_2
Merge made by the 'ort' strategy.
 1.txt | 2 +-
 3.txt | 2 +-
 git   | 0
 3 files changed, 2 insertions(+), 2 deletions(-)
 create mode 100644 git
```

Рисунок 17.1 – Слияние двух веток

18. Решаю конфликт файла 1.txt в ручном режиме, а конфликт 3.txt используя команду git mergetool с помощью одной из доступных утилит, например Meld.

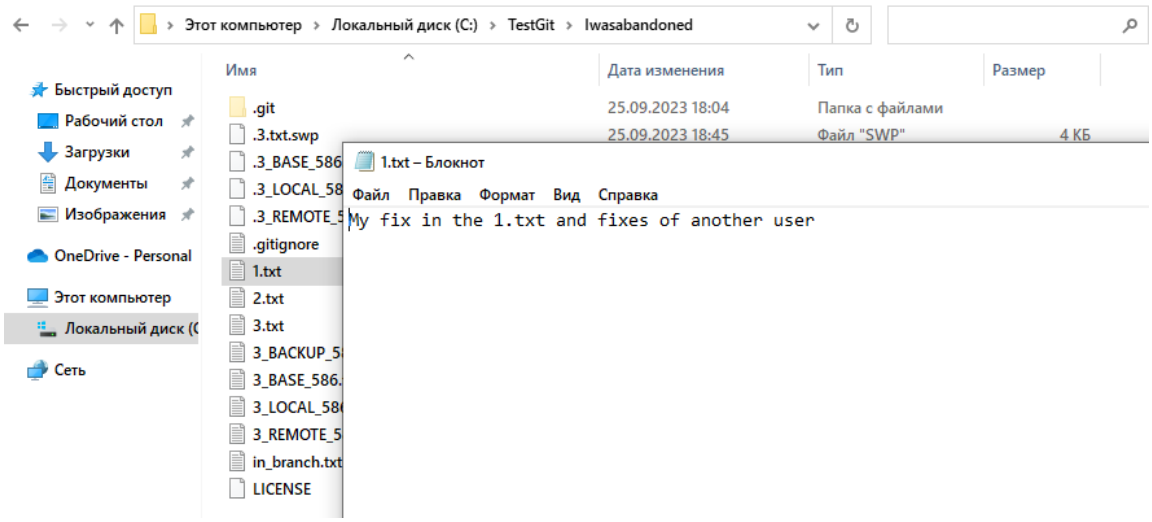

Рисунок 18.1 – Устранение конфликта в файле 1.txtвручную

```
C:\TestGit\Iwasabandoned>git status
On branch branch_1
Your branch is up to date with 'origin/branch_1'.

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Changes to be committed:
        modified:   1.txt

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   3.txt
```

Рисунок 18.2 – Конфликт в файле 1.txtустранён

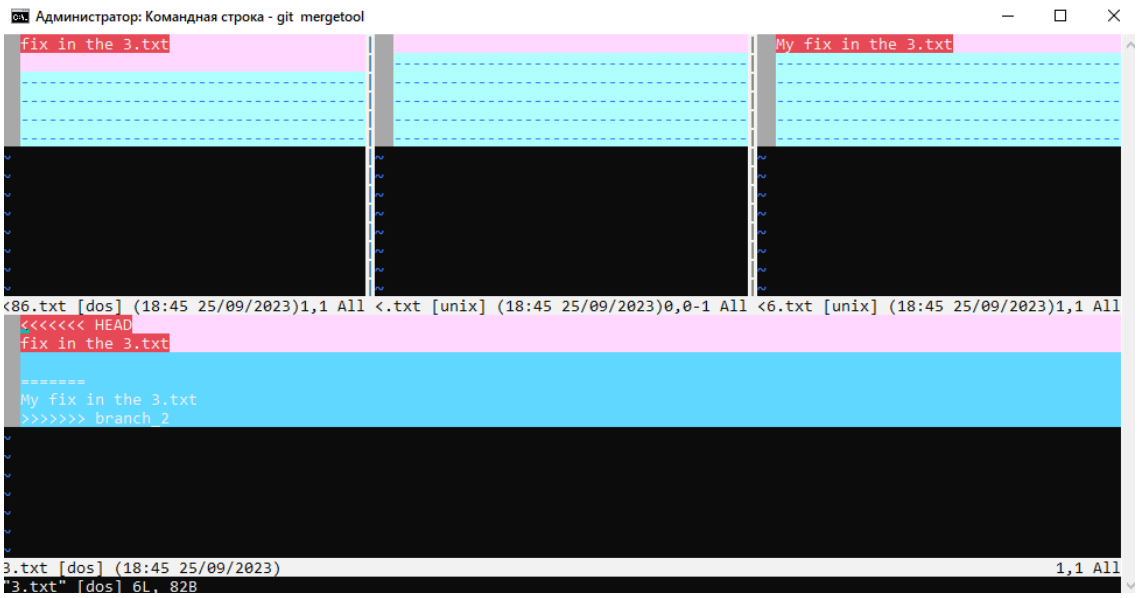

Рисунок 18.3 – Устранение конфликта файла 3.txtс помощью инструмента mergetool

```
C:\git1\lab3>git add 3.txt

C:\git1\lab3>git commit -m"file 3.txt was fixed"
On branch branch_1
nothing to commit, working tree clean
```

Рисунок 18.4 – Оба конфликты устранены

19. Отправляю ветку branch_1 на GitHub

```
C:\git1\lab3>git push origin branch_1
Enumerating objects: 30, done.
Counting objects: 100% (30/30), done.
Delta compression using up to 2 threads
Compressing objects: 100% (20/20), done.
Writing objects: 100% (29/29), 2.54 KiB | 325.00 KiB/s, done.
Total 29 (delta 11), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (11/11), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:      https://github.com/mryadskaya/lab3/pull/new/branch_1
remote:
To https://github.com/mryadskaya/lab3.git
 * [new branch]      branch_1 -> branch_1

C:\git1\lab3>git pull
```

Рисунок 19.1 – Отправление ветки branch_1 на сервер

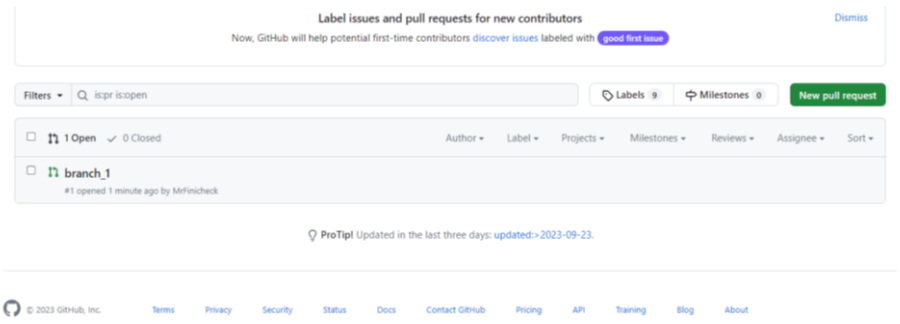

Рисунок 19.2 – Наша ветка branch_1 в репозитории GitHub

20. Создаю средствами GitHub удаленную ветку branch_3

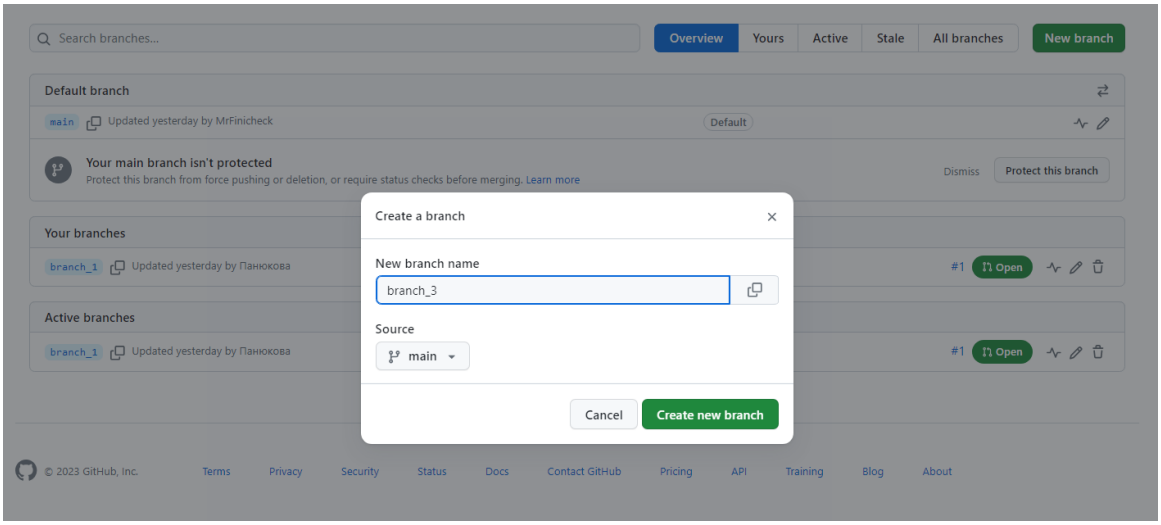

Рисунок 20.1 – Создание ветки branch_3 в репозитории

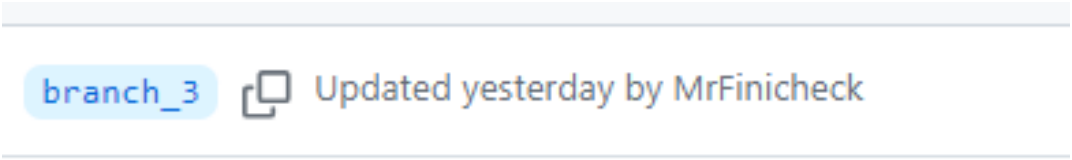

Рисунок 20.2 – Наша созданная удаленная ветка

21. Создать в локальном репозитории ветку отслеживания удаленной ветки branch_3

```
C:\git1\lab3>git pull
From https://github.com/mryadskaya/lab3
 * [new branch]      branch-3   -> origin/branch-3
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=origin/<branch> branch_1


C:\git1\lab3>git branch --all
* branch_1
  branch_2
  main
  new_m
  remotes/origin/HEAD -> origin/main
  remotes/origin/branch-3
  remotes/origin/branch_1
  remotes/origin/main
```

Рисунок 21.1 – Получаем данные с удаленного сервера с помощью команды gitpull

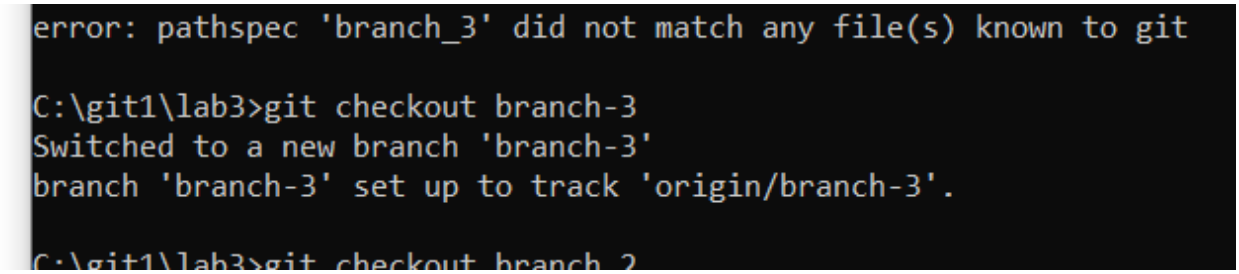

Рисунок 21.2 – С помощью команды gitcheckoutсоздаем ветку отслеживания удаленной ветки

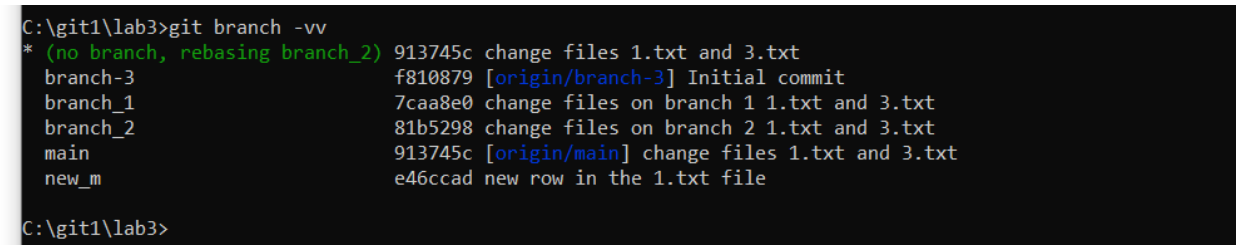

Рисунок 21.3 – Ветка branch_3 отслеживает удалённую

22. Переходим на ветку branch_3 и добавляемвфайл 2.txt строку "the final fantasy in the 4.txt file"

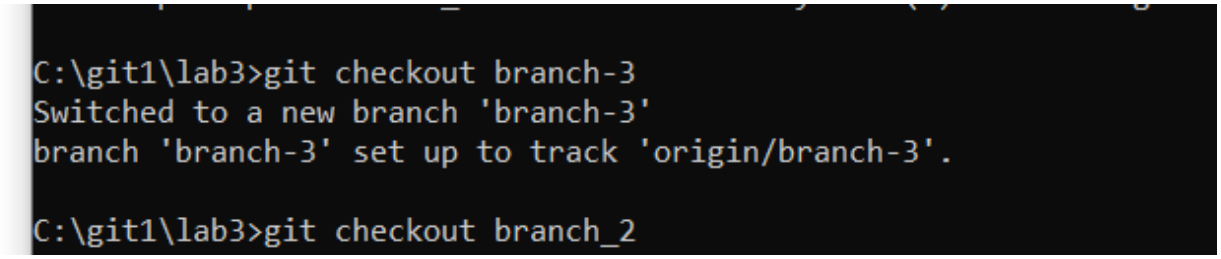

Рисунок 22.1 – Переход на ветку branch_3

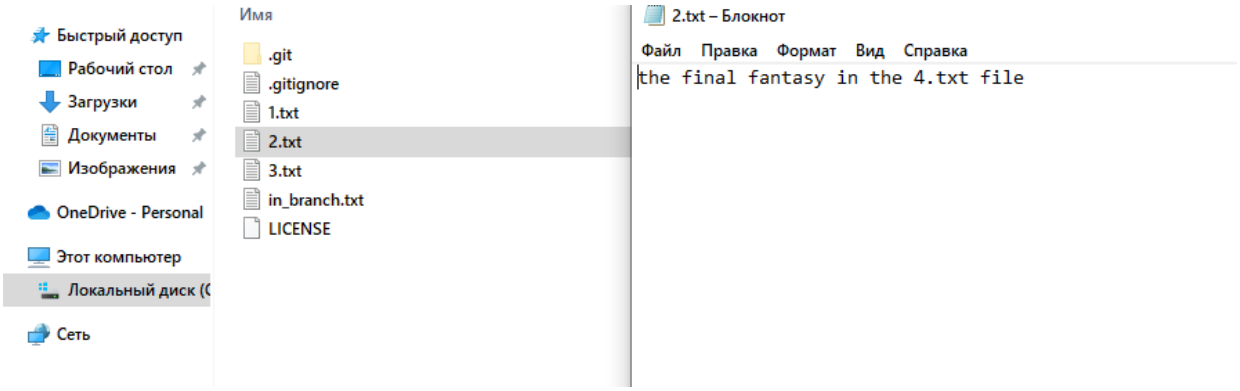

Рисунок 22.2 – Добавление в файл 2.txtновой строки

23. Выполняю перемещение ветки main на ветку branch_2

```
C:\git1\lab3>git checkout branch_2
Switched to branch 'branch_2'
```

Рисунок 23.1 – Переходим на ветку branch_2

```
C:\git1\lab3>git rebase main
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Auto-merging 3.txt
CONFLICT (content): Merge conflict in 3.txt
error: could not apply 66ee89a... change files on branch 1 1.txt and 3.txt
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
Could not apply 66ee89a... change files on branch 1 1.txt and 3.txt
```

Рисунок 23.2 – Перемещаемветку mainна ветку branch_2 с помощью команды gitrebase

24. Отправляю изменения веток main и branch_2 на GitHub

```
C:\git1\lab3>git push origin branch_2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:      https://github.com/mryadskaya/lab3/pull/new/branch_2
remote:
To https://github.com/mryadskaya/lab3.git
 * [new branch]      branch_2 -> branch_2
```

Рисунок 24.1 – Отправление ветки branch_2 на GitHub

```
C:\git1\lab3>git push --set-upstream origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 364 bytes | 91.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/mryadskaya/lab3.git
   f810879..913745c  main -> main
branch 'main' set up to track 'origin/main'.
```

Рисунок 24.2 – Отправление изменений ветки maim

https://github.com/mryadskaya/lab3.git

Контрольные вопросы

1. Что такое ветка?

В Git ветки — это элемент повседневного процесса разработки. По сути ветки в Git представляют собой указатель на снимок изменений. Если нужно добавить новую возможность или исправить ошибку Вы создаете новую ветку, в которой будут размещаться эти изменения.Используя ветвление, Вы отклоняетесь от основной линии разработки и продолжаете работу независимо от неё, не вмешиваясь в основную линию.

2. Что такое HEAD?

Во-первых, HEAD – это указатель на коммит в вашем репозитории, который станет родителем следующего коммита. Во-вторых, HEAD указывает на коммит, относительного которого будет создана рабочая копия во время операции checkout. Другими словами, когда вы переключаетесь с ветки на ветку, используя операцию checkout, то в вашем репозитории указатель HEAD будет переключаться между последними коммитами выбираемых вами ветвей.

3. Способы создания веток

Вы можете это сделать командой git branch. Чтобы создать ветку и сразу переключиться на нее, можно выполнить команду git checkout с параметром -b

4. Как узнать текущую ветку?

С помощью команды git branch --all

5. Как переключаться между ветками?

git checkout <название_ветки>

6. Что такое удаленная ветка?

Удалённые ветки — это ссылки на состояние веток в ваших удалённых репозиториях.

7. Что такое ветка отслеживания?

Ветки слежения — это локальные ветки, которые напрямую связаны с удалённой веткой

8. Как создать ветку отслеживания?

С помощью команды gitcheckout -b /<название_ветки>origin/<название_ветки>

9. Как отправить изменения из локальной ветки в удаленную ветку?

С помощью команды gitpush

10. В чем отличие команд git fetch и git pull?

Команда git fetch получает с сервера все изменения, которых у вас ещё нет, но не будет изменять состояние вашей рабочей директории. Эта команда просто получает данные и позволяет вам самостоятельно сделать слияние. Тем не менее, существует команда git pull, которая в большинстве случаев является командой git fetch , за которой непосредственно следует команда git merge. Если у вас настроена ветка слежения как показано в предыдущем разделе, или она явно установлена, или она была создана автоматически командами clone или checkout, git pull определит сервер и ветку, за которыми следит ваша текущая ветка, получит данные с этого сервера и затем попытается слить удалённую ветку

11. Как удалить локальную и удаленную ветки?

Для удаления ветки на сервере нужно выполнить команду git push origin --delete <название_ветки>. Для локального удаления ветки выполните команду git branch с параметром -d

12. Изучить модель ветвления git-flow (использовать материалы статей https://www.atlassian.c om/ru/git/tutorials/comparing-workflows/gitflow-workflow, https://habr.com/ru/post/10691 2/). Какие основные типы веток присуствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?

В модели git-flow есть две главных ветки (master и develop), а также несколько дополнительных.

Последовательность действий при работе по модели gitflow:

1. Из ветки main создается ветка develop
2. Из ветки develop создается ветка release.
3. Из ветки develop создаются ветки feature.
4. Когда работа над веткой feature завершается, она сливается в ветку develop.
5. Когда работа над веткой release завершается, она сливается с ветками develop и main.
6. Если в ветке main обнаруживается проблема, из main создается ветка hotfix.
7. Когда работа над веткой hotfix завершается, она сливается с ветками develop и main.

Основные недостатки: git-flow изначально сложна и запутана, с git-flow потенциальное количество merge-конфликтов теперь минимум в три раза выше, сторонникам git-flow придется отказаться от перебазирования: ведь оно происходит вместе со слиянием, в результате которого две ветви объединяются.

13. На прошлой лабораторной работе было задание выбрать одно из программных средств с GUI для работы с Git. Необходимо в рамках этого вопроса привести описание инструментов для работы с ветками Git, предоставляемых этим средством.

Чтобы создать новую ветку в GitHub Desktop переходим в Branch> New Branch и создаем новую ветвь. Даём ей имя и нажимаем Create Branch.После создания ветки, в центре раскрывающееся меню будет указывать на ту ветку, в которой мы работаем.