

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 №3
дисциплины «Основы программной инженерии»

Выполнила:
Ламская Ксения Вячеславовна
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. Изучила теоретический материал работы.
2. Создала общедоступный репозиторий на GitHub.

Create a new repository



A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *	Repository name *
 ksenia-lamskaya ▾	/ 3laba
✔ 3laba is available.	

Great repository names are short and memorable. Need inspiration? How about **verbose-enigma** ?

Description (optional)

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

- ☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

Рисунок 2.1 – Настройка репозитория

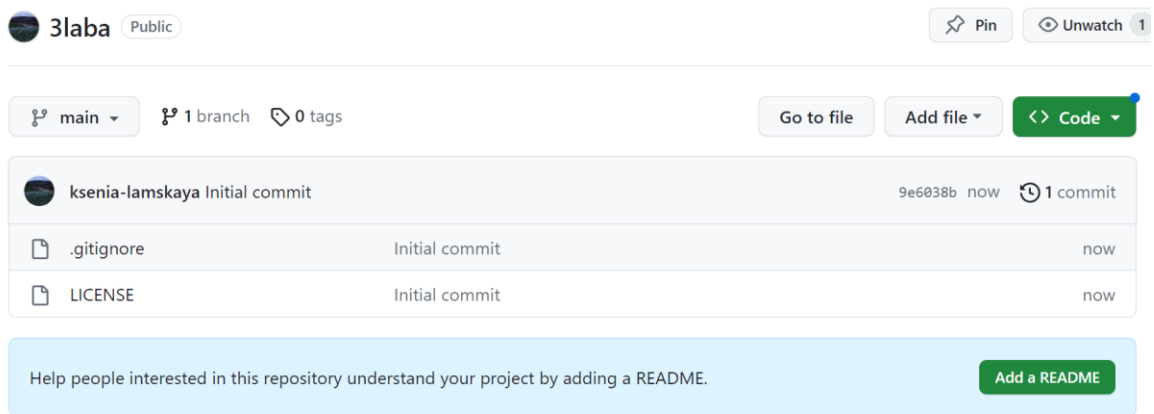


Рисунок 2.2 – Готовый репозиторий

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

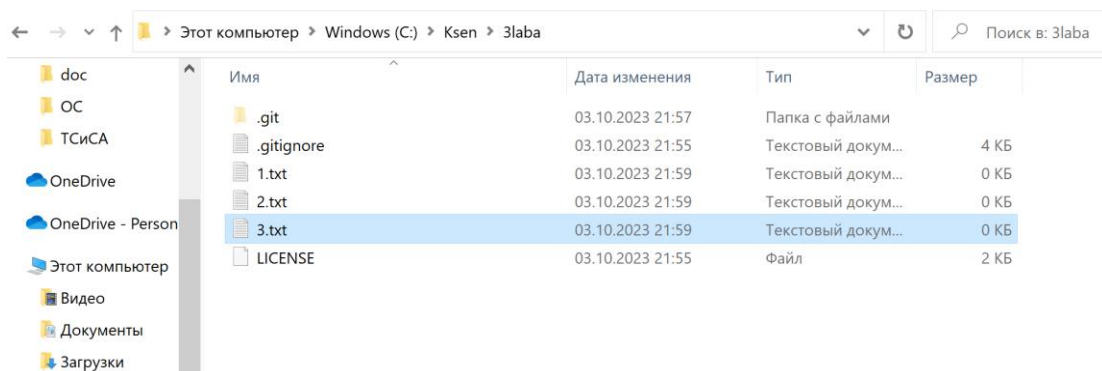


Рисунок 3.1 – Папка репозитория

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

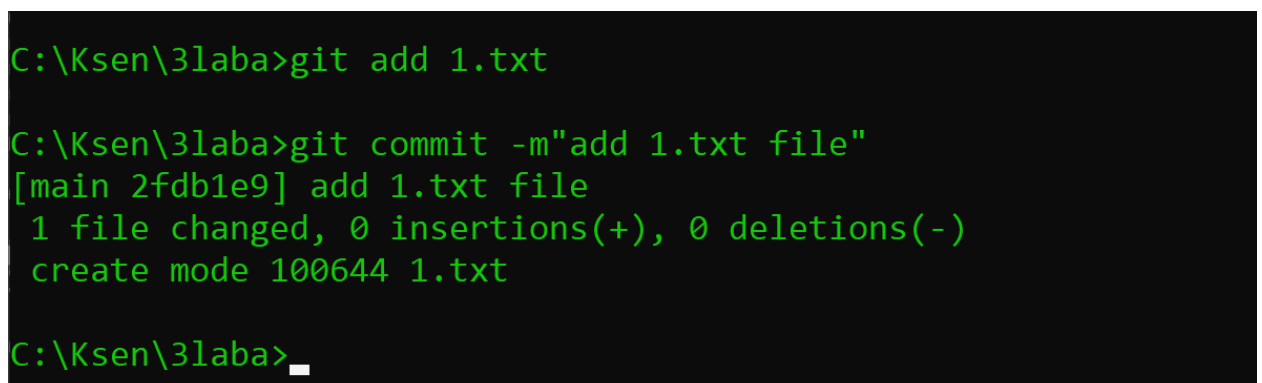


Рисунок 4.1 – Индексирование и коммит файла 1.txt

5. Проиндексировала второй и третий файлы.

```
C:\Ksen\3laba>git add 2.txt 3.txt  
C:\Ksen\3laba>_
```

Рисунок 5.1 – Индексация файлов 2.txt и 3.txt

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

```
C:\Ksen\3laba>git commit -m"add 2.txt and 3.txt"  
[main 4468aa1] add 2.txt and 3.txt  
2 files changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 2.txt  
create mode 100644 3.txt
```

Рисунок 6.1 – Коммит файлов 2.txt и 3.txt

7. Создала новую ветку my_first_branch

```
C:\Ksen\3laba>git branch my_first_branch  
  
C:\Ksen\3laba>git log --online --decorate  
fatal: unrecognized argument: --online  
  
C:\Ksen\3laba>git log --oneline --decorate  
4468aa1 (HEAD -> main, my_first_branch) add 2.txt and 3.txt  
2fdb1e9 add 1.txt file  
0b508ff (origin/main, origin/HEAD) d  
932bf5c kkk  
9e6038b Initial commit
```

Рисунок 7.1 – Создание ветки

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

```
C:\Ksen\3laba>git checkout my_first_branch  
Switched to branch 'my_first_branch'
```

Рисунок 8.1 – Переход на ветку my_first_branch

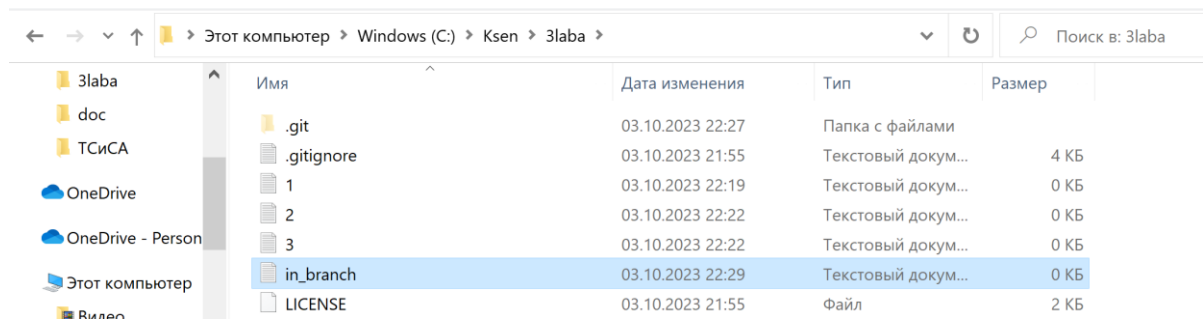


Рисунок 8.2 – Создание файла in_branch.txt

```
C:\Ksen\3laba>git add .  
  
C:\Ksen\3laba>git commit -m"commit on branch"  
[my_first_branch cea765b] commit on branch  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 in_branch.txt  
  
C:\Ksen\3laba>
```

Рисунок 8.3 – Коммит изменений

9. Возвращаюсь на ветку main

```
C:\Ksen\3laba>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)
```

Рисунок 9.1 – Переход на главную ветку

10. Создаю и сразу перехожу на ветку new_branch

```
C:\Ksen\3laba>git checkout -b new_branch  
Switched to a new branch 'new_branch'
```

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

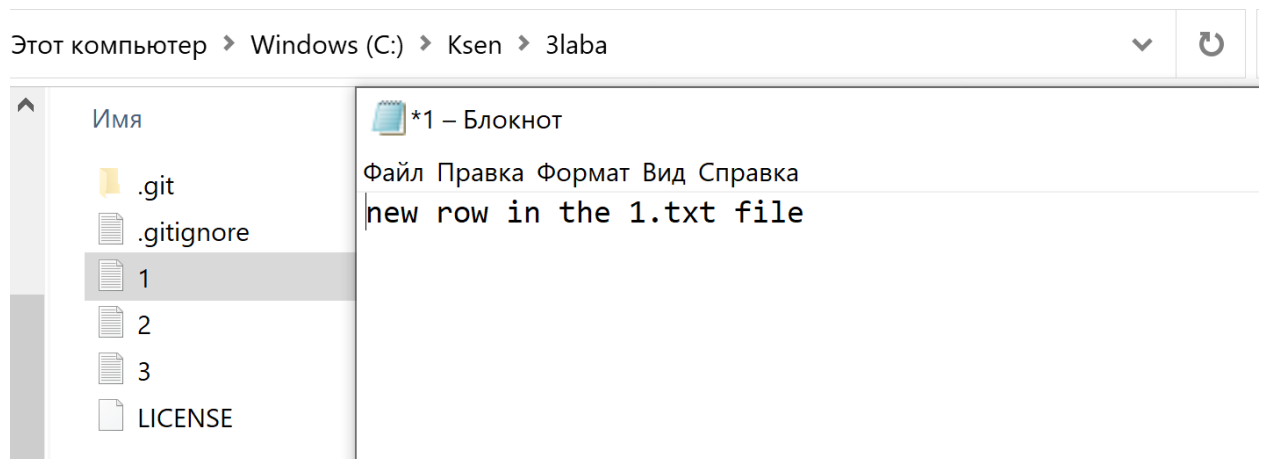


Рисунок 11.1 – Изменение файла 1.txt добавлением строки

```
C:\Ksen\3laba>git add .  
  
C:\Ksen\3laba>git commit -m"changes in 1.txt file"  
[new_branch dedc056] changes in 1.txt file  
1 file changed, 1 insertion(+)
```

Рисунок 11.2 – Коммит изменений

12. Перейти на ветку main и слить ветки master и my_first_branch, после чего слить ветки main и new_branch

```
C:\Ksen\3laba>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:\Ksen\3laba>git merge my_first_branch  
Updating 4468aa1..cea765b  
Fast-forward  
in_branch.txt | 0  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 in_branch.txt
```

Рисунок 12.1 – Переход на ветку main и слияние с веткой my_first_branch

```
C:\Ksen\3laba>git merge new_branch  
Merge made by the 'ort' strategy.  
1.txt | 1 +  
1 file changed, 1 insertion(+)
```

Рисунок 12.2 – Слияние с веткой new_branch

13. Удаляю ветки my_first_branch и new_branch

```
C:\Ksen\3laba>git branch -d my_first_branch
Deleted branch my_first_branch (was cea765b).

C:\Ksen\3laba>git branch -d new_branch
Deleted branch new_branch (was dedc056).
```

Рисунок 13.1 – Удаление ненужных веток

14. Создаю ветки branch_1 и branch_2

```
C:\Ksen\3laba>git branch branch_1

C:\Ksen\3laba>git branch branch_2

C:\Ksen\3laba>git log --oneline --decorate
c0b4510 (HEAD -> main, branch_2, branch_1) Merge branch 'new_branch'
dedc056 changes in 1.txt file
cea765b commit on branch
4468aa1 add 2.txt and 3.txt
2fdb1e9 add 1.txt file
0b508ff (origin/main, origin/HEAD) d
932bf5c kkk
9e6038b 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.

```
C:\Ksen\3laba>git checkout branch_1
Switched to branch 'branch_1'
```

Рисунок 14.2 – Перехожу на ветку branch_1

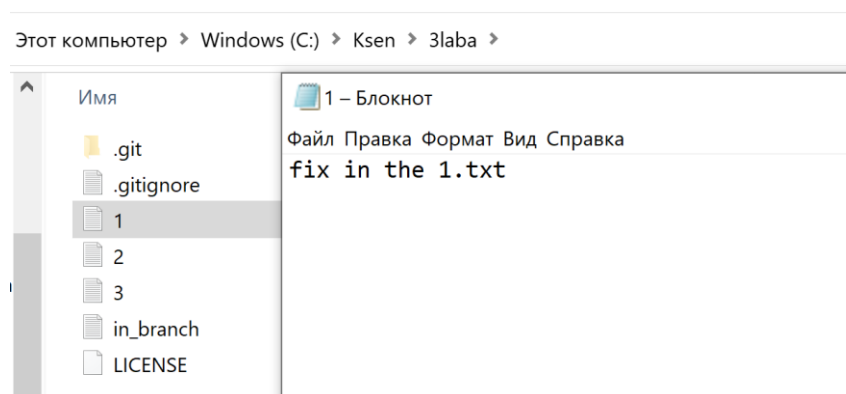


Рисунок 15.1 – Изменяю файл 1.txt, удаляю все содержимое и добавляя текст

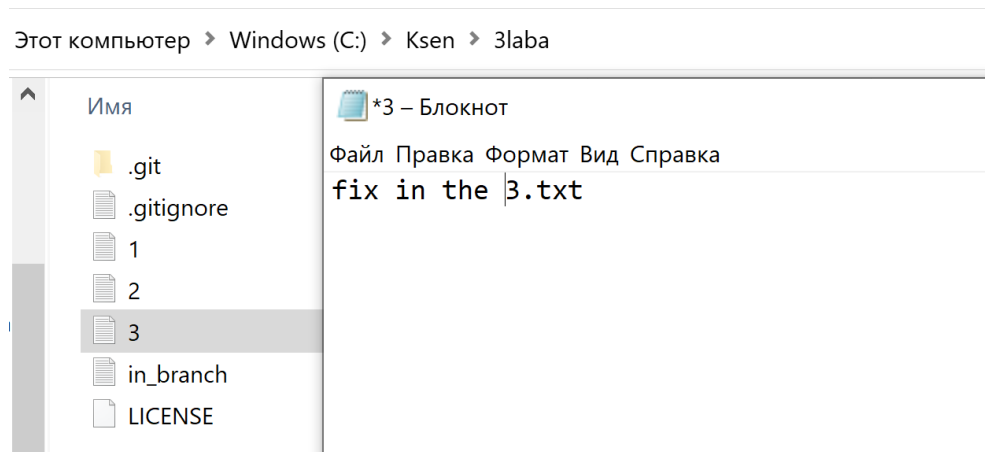


Рисунок 15.2 – Изменяю файл 3.txt, удаляю все содержимое и добавляя текст

```
C:\Ksen\3laba>git add .  
C:\Ksen\3laba>git commit -m"change files 1.txt and 3.txt"  
[branch_1 99876ec] change files 1.txt and 3.txt  
2 files changed, 2 insertions(+), 1 deletion(-)
```

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:\Ksen\3laba>git checkout branch_2  
Switched to branch 'branch_2'
```

Рисунок 16.1 – Переход на другую ветку

Этот компьютер > Windows (C:) > Ksen > 3laba

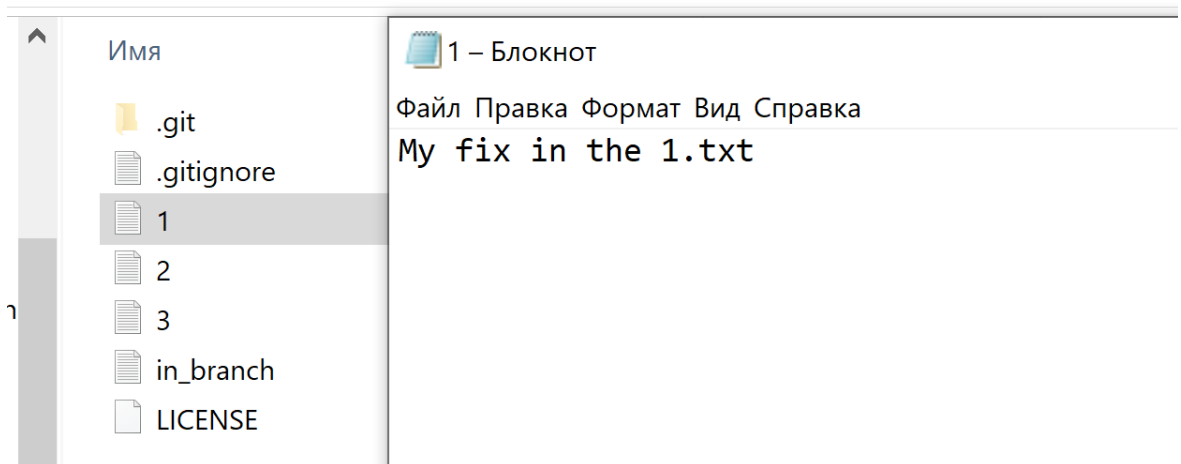


Рисунок 16.2 – Изменяю файл 1.txt, удаляя все содержимое и добавляя текст

Этот компьютер > Windows (C:) > Ksen > 3laba >

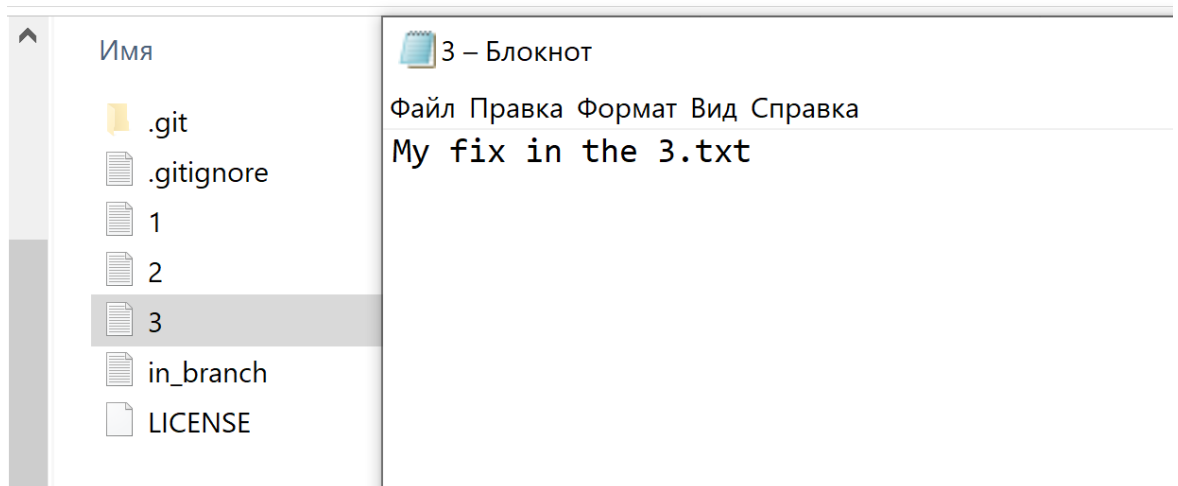


Рисунок 16.3 – Изменяю файл 3.txt, удаляя все содержимое и добавляя текст

```
C:\Ksen\3laba>git add .  
  
C:\Ksen\3laba>git commit -m"change files 1.txt and 3.txt on branch_2"  
[branch_2 8648378] change files 1.txt and 3.txt on branch_2  
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 16.4 – Коммит изменений

17. Сливаю изменения ветки branch_2 в ветку branch_1

```
C:\Ksen\3laba>git checkout branch_1
Switched to branch 'branch_1'

C:\Ksen\3laba>git merge branch_2
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Auto-merging 3.txt
CONFLICT (content): Merge conflict in 3.txt
Automatic merge failed; fix conflicts and then commit the result.
```

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.



*1 – Блокнот

Файл Правка Формат Вид Справка

My fix in the 1.txt and fixin the 1.txt

My fixes in the 1.txt|

Рисунок 18.1 – Устранение конфликта в файле 1.txt вручную

```
C:\Ksen\3laba>git add 1.txt

C:\Ksen\3laba>git status
On branch 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.txt устранён

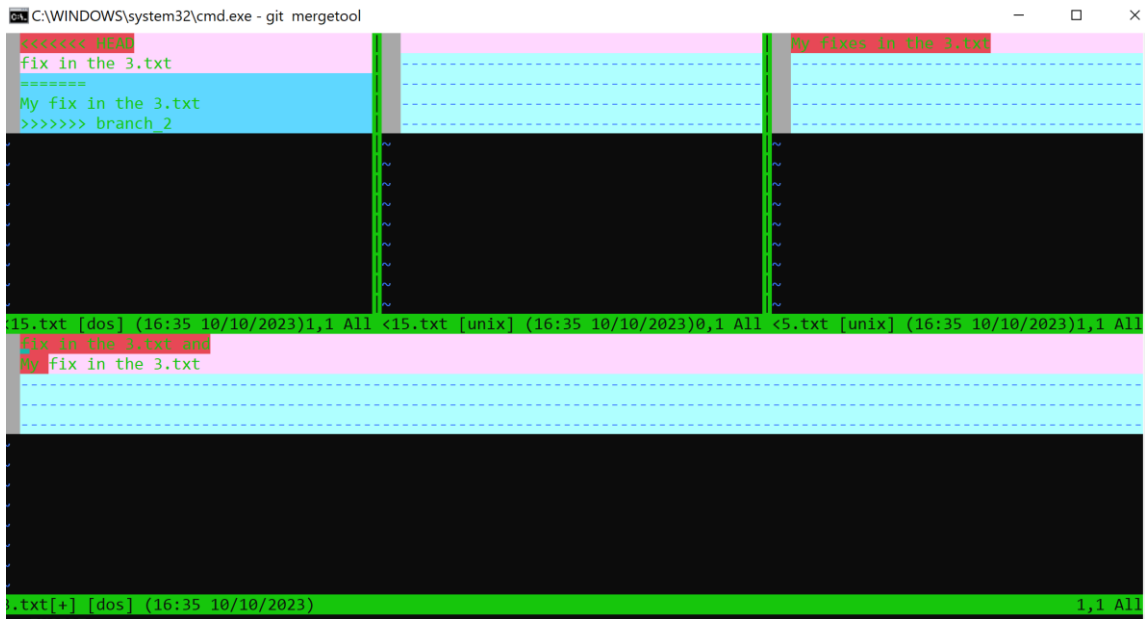


Рисунок 18.3 – Устранение конфликта файла 3.txt с помощью инструмента mergetool

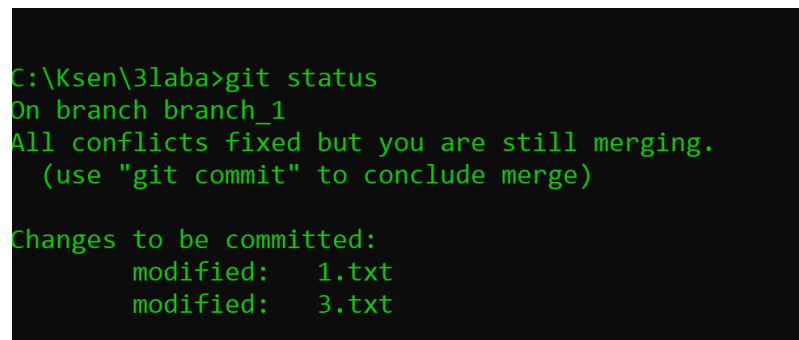


Рисунок 18.4 – Оба конфликта устранены

19. Отправляю ветку branch_1 на GitHub

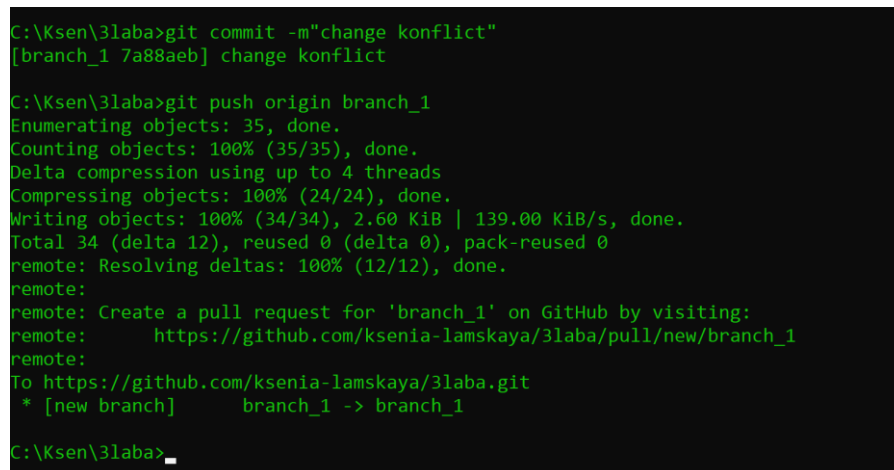


Рисунок 19.1 – Отправление ветки branch_1 на сервер

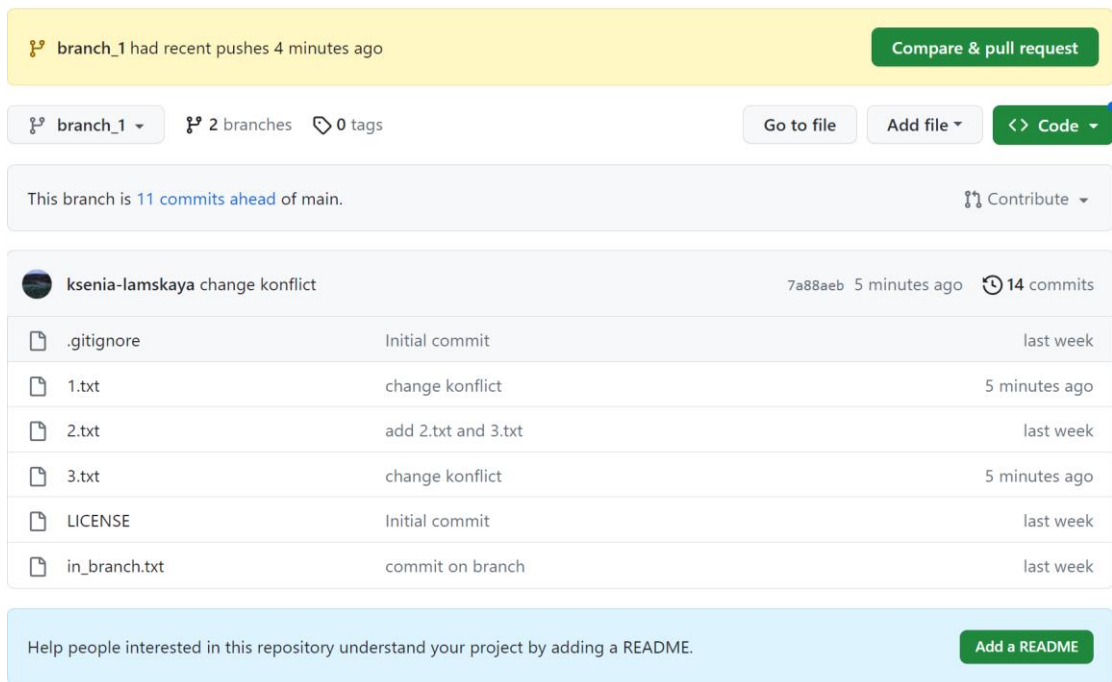


Рисунок 19.2 – Наша ветка branch_1 в репозитории GitHub

20. Создаю средствами GitHub удаленную ветку branch_3

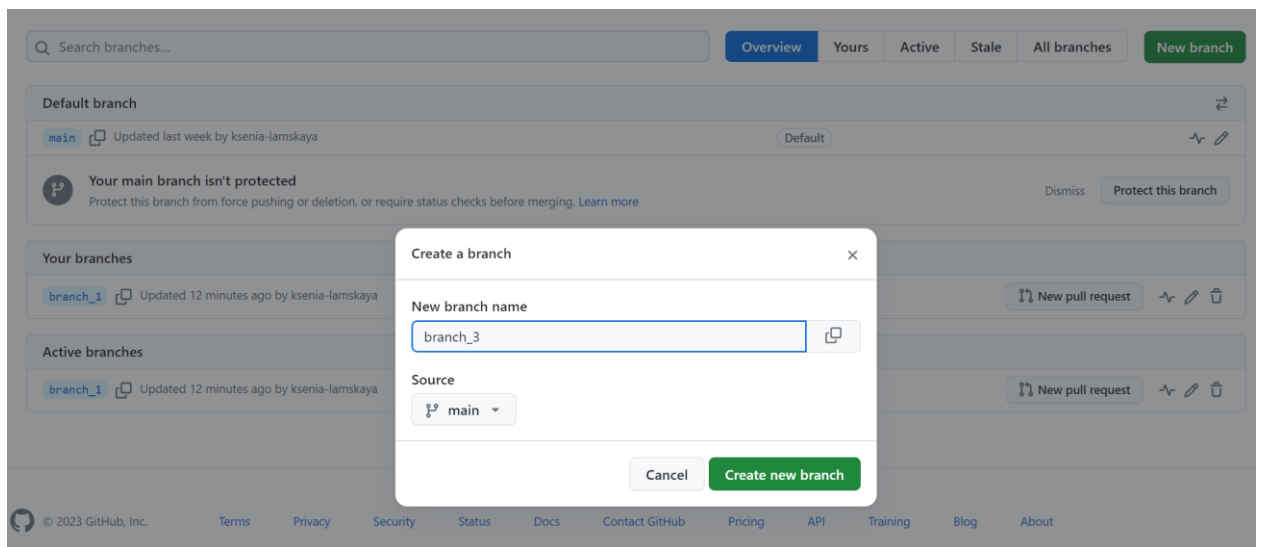


Рисунок 20.1 – Создание ветки branch_3 в репозитории

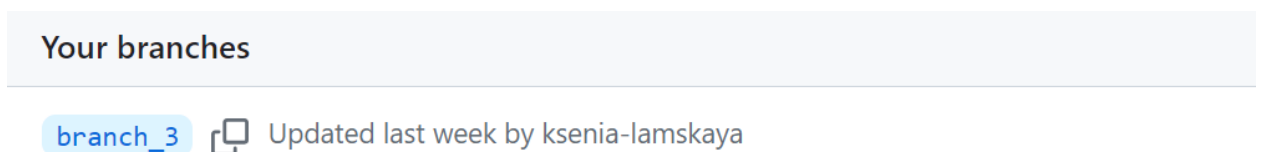


Рисунок 20.2 – Наша созданная удаленная ветка

21. Создать в локальном репозитории ветку отслеживания удаленной ветки branch_3

```
C:\Ksen\3laba>git pull
From https://github.com/ksenia-lamskaya/3laba
* [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:\Ksen\3laba>git branch --all
* branch_1
  branch_2
  main
remotes/origin/HEAD -> origin/main
remotes/origin/branch_1
remotes/origin/branch_3
remotes/origin/main
C:\Ksen\3laba>_
```

Рисунок 21.1 – Получаем данные с удаленного сервера с помощью команды git pull

```
C:\Ksen\3laba>git checkout branch_3
Switched to a new branch 'branch_3'
branch 'branch_3' set up to track 'origin/branch_3'.
```

Рисунок 21.2 – С помощью команды git checkout создаем ветку отслеживания удаленной ветки

```
C:\Ksen\3laba>git branch -vv
branch_1 7a88aeb change konflikt
branch_2 d56007b add my fixes in 1.txt and 3.txt
* branch_3 0b508ff [origin/branch_3] d
main     c0b4510 [origin/main: ahead 5] Merge branch 'new_branch'
```

Рисунок 21.3 – Ветка branch_3 отслеживает удалённую

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

```
C:\Ksen\3laba>git checkout branch_3
Already on 'branch_3'
Your branch is up to date with 'origin/branch_3'.
```

Рисунок 22.1 – Переход на ветку branch_3

2 – Блокнот

Файл Правка Формат Вид Справка

"the final fantasy in the 4.txt file"

Рисунок 22.2 – Добавление в файл 2.txt новой строки

```
C:\Ksen\3laba>git add .  
  
C:\Ksen\3laba>git commit -m"add file 2.txt"  
[branch_3 cfe8325] add file 2.txt  
1 file changed, 1 insertion(+)  
create mode 100644 2.txt  
  
C:\Ksen\3laba>git push  
Enumerating objects: 4, done.  
Counting objects: 100% (4/4), done.  
Delta compression using up to 4 threads  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (3/3), 335 bytes | 111.00 KiB/s, done.  
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
To https://github.com/ksenia-lamskaya/3laba.git  
0b508ff..cfe8325 branch_3 -> branch_3
```

23. Выполняю перемещение ветки main на ветку branch_2

```
C:\Ksen\3laba>git checkout branch_2  
Switched to branch 'branch_2'
```

Рисунок 23.1 – Переходим на ветку branch_2

```
C:\Ksen\3laba>git rebase main  
Current branch branch_2 is up to date.
```

Рисунок 23.2 – Перемещаем ветку main на ветку branch_2 с помощью команды git rebase

24. Отправляю изменения веток main и branch_2 на GitHub

```
C:\Ksen\3laba>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/ksenia-lamskaya/3laba/pull/new/branch_2  
remote:  
To https://github.com/ksenia-lamskaya/3laba.git  
* [new branch] branch_2 -> branch_2
```

Рисунок 24.1 – Отправление ветки branch_2 на GitHub

```
C:\Ksen\3laba>git push --set-upstream origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ksenia-lamskaya/3laba.git
   0b508ff..c0b4510  main -> main
branch 'main' set up to track 'origin/main'.
```

Рисунок 24.2 – Отправление изменений ветки main

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, относительно 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 `git checkout -b /<название_ветки> origin/<название_ветки>`

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

С помощью команды `git push`

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 удалённую 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.com/ru/git/tutorials/comparing-workflows/gitflow-workflow>, <https://habr.com/ru/post/106912/>). Какие

основные типы веток присутствуют 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.