

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»

Выполнил:
Матвеев Александр Иванович
1 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: основы ветвления Git.

Цель работы: исследование базовых возможностей по работе с локальными и удаленными ветками Git.

Ход работы.

1. Создание нового репозитория с лицензией MIT.

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 * SashkaHacker / **Repository name *** laba3
✔ laba3 is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-happiness](#) ?

Description (optional)
Выполнение лабораторной работы №3 по дисциплине: основы программной инженерии

☒ **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.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

ⓘ You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание репозитория

2. Выполнение пунктов 3-6.

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git add 1.txt

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git commit -m "add 1.txt file"
[main e28643c] add 1.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
```

Рисунок 2 – Индексирование 1.txt и коммит “add 1.txt file”

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git add .

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git commit --amend -m "add 2.txt and 3.txt"
[main 3040839] add 2.txt and 3.txt
Date: Wed Sep 27 07:24:57 2023 +0300
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
create mode 100644 2.txt
create mode 100644 3.txt
```

Рисунок 3 – Добавление в индекс 2.txt и 3.txt, изменение коммита

3. Создание новой ветки.

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git checkout -b my_first_branch
Switched to a new branch 'my_first_branch'
```

Рисунок 4 – Создание ветки “my_first_branch”

4. Создание в новой ветке коммита.

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (my_first_branch)
$ git add .

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (my_first_branch)
$ git commit -m "add file in_branch.txt in new branch"
[my_first_branch ffae89c] add file in_branch.txt in new branch
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt
```

Рисунок 5 – Коммит файла в новой ветке

5. Выполнение заданий 9-10.

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (my_first_branch)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git checkout -b new_branch
Switched to a new branch 'new_branch'
```

Рисунок 6 – Создание и мгновенный переход на ветку new_branch

6. Изменение содержимого файла 1.txt, коммит в ветке new_branch.

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (new_branch)
$ git add 1.txt

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (new_branch)
$ git commit -m "add changes in file 1.txt"
[new_branch 0c5590a] add changes in file 1.txt
1 file changed, 1 insertion(+)
```

Рисунок 6 – Коммит изменений

7. Добавил в файл README.md необходимую информацию.

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (new_branch)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git merge my_first_branch
Updating 3040839..ffae89c
Fast-forward
 in_branch.txt | 0
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git merge new_branch
Merge made by the 'ort' strategy.
1.txt | 1 +
1 file changed, 1 insertion(+)
```

Рисунок 7 – Слияние веток my_first_branch и new_branch с веткой main

8. Удаление веток new_branch и my_first_branch.

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git branch -d new_branch
Deleted branch new_branch (was 0c5590a).

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git branch -d my_first_branch
Deleted branch my_first_branch (was ffae89c).
```

Рисунок 8 – Удаление ненужных веток

9. Выполнение заданий 14-19.

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git branch branch_1

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git branch branch_2
```

Рисунок 9 – Создание веток branch_1 и branch_2

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (main)
$ git checkout branch_1
Switched to branch 'branch_1'

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (branch_1)
$ add .
bash: add: command not found

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (branch_1)
$ git add .

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (branch_1)
$ git commit -m "add changes in file 1.txt and 3.txt"
[branch_1 c4bb90a] add changes in file 1.txt and 3.txt
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 10 – Результат выполнения задания 15

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (branch_1)
$ git checkout branch_2
Switched to branch 'branch_2'

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (branch_2)
$ git add .

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (branch_2)
$ git commit -m "add changes in file 1.txt and 3.txt"
[branch_2 0ce892d] add changes in file 1.txt and 3.txt
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 11 – Результат выполнения задания 16

```
$ git checkout branch_1
Switched to branch 'branch_1'

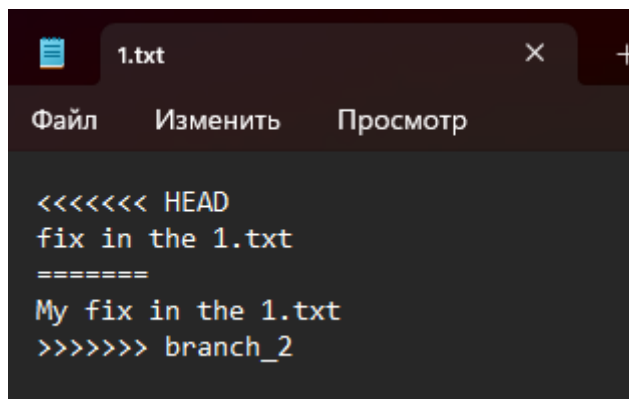
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (branch_1)
$ git merge branch_2
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Auto-merging 3.txt
CONFLICT (content): Merge conflict in 3.txt
Automatic merge failed; fix conflicts and then commit the result.

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (branch_1|MERGING)
$ git status
On branch branch_1
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   1.txt
    both modified:   3.txt

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

Рисунок 12 – Слияние ветки branch_1 с веткой branch_2



```
1.txt
Файл  Изменить  Просмотр

<<<<<< HEAD
fix in the 1.txt
=====
My fix in the 1.txt
>>>>>> branch_2
```

Рисунок 13 – Решение конфликта в файле 1.txt

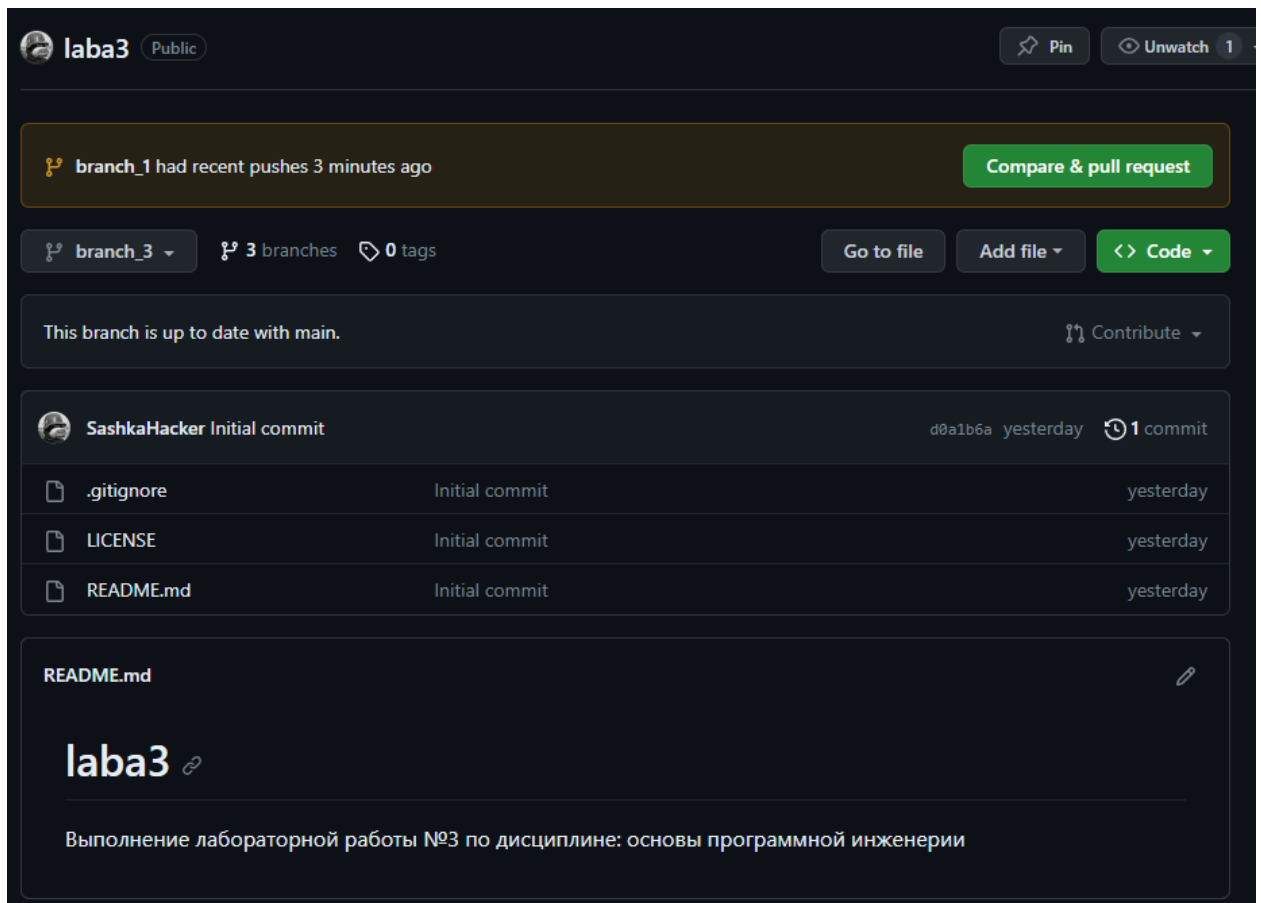


Рисунок 16 – branch_3

11. Добавление файла с текстом в ветке branch_3.

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (branch_3)
$ git add .

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (branch_3)
$ git commit -m "add file 2.txt"
[branch_3 321d8f2] add file 2.txt
1 file changed, 1 insertion(+)
create mode 100644 2.txt
```

Рисунок 17 – Коммит

12. Выполнение перемещения ветки master на ветку branch_2, а также отправление этих веток на сервер.


```

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (branch_3)
$ git checkout master
Switched to a new branch 'master'
branch 'master' set up to track 'origin/master'.

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (master)
$ git merge branch_2
Updating d0a1b6a..0ce892d
Fast-forward
 1.txt      | 1 +
 2.txt      | 0
 3.txt      | 1 +
 in_branch.txt | 0
4 files changed, 2 insertions(+)
create mode 100644 1.txt
create mode 100644 2.txt
create mode 100644 3.txt
create mode 100644 in_branch.txt

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (master)
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/SashkaHacker/laba3.git
   d0a1b6a..0ce892d  master -> master

Sashka@DESKTOP-U4RPSBI MINGW64 ~/git/laba3 (master)
$ 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/SashkaHacker/laba3/pull/new/branch_2
remote:
To https://github.com/SashkaHacker/laba3.git
 * [new branch]      branch_2 -> branch_2

```

Рисунок 18 – Синхронизация веток

Контрольные вопросы:

1. В git ветки представляют собой отдельные линии разработки. Каждая ветка содержит свою версию проекта. Они нужны для изоляции работы над какими-либо функциями приложения или фикса багов.
2. HEAD – является указателем на текущий коммит в выбранной ветке.
3. Ветку можно создать различными способами: (`git branch nameOfBranch`) – создание новой ветки, но не переключение на нее. (`git checkout -b nameOfBranch`) – создание новой ветки и мгновенное переключение на нее.
4. Для того, чтобы узнать текущую ветку, нужно использовать команду (`git branch`), она выведет весь список веток в репозитории, текущая ветка будет отмечена символом *. Также можно использовать команду (`git branch -a`), она выведет как локальные, так и удаленные ветки.
5. Для переключения между ветками нужно использовать команду (`git checkout nameOfBranch`).
6. Удаленные ветки — это ветки, которые существуют на удаленном репозитории.
7. Ветки отслеживания — это локальные ветки, которые напрямую связаны с удалённой веткой.
8. Существует различное множество способов создания ветки отслеживания. Например при помощи команды (`git checkout <branchName> <remote>/<branch>`), но существует сокращение для данной команды (`git checkout --track <remote>/<branch>`). Но даже для этого сокращения существует свое сокращение: (`git checkout <NameRemoteBranch>`).
9. С помощью команды «`git push <remote> <branchName>`».

10. Команда (`git fetch`) получает с сервера все изменения, но не будет делать автоматическое слияние, (`git pull`) в свою очередь делает автоматическое слияние, когда получает изменения с сервера.

11. Для удаления локальной ветки необходимо воспользоваться командой (`git branch -d <branchName>`), для удаления удаленной ветки необходимо воспользоваться командой (`git push <remote> --delete <branchName>`).

12. Модель `git-flow` предлагает основные типы веток для организации разработки в Git. В ней присутствуют следующие типы веток: Master (главная ветка). Develop (ветка разработки). Feature (фичевая ветка). Release (ветка релиза). Hotfix (ветка исправления).

1) Новая функциональность разрабатывается в отдельной ветке Feature, которая создается от ветки Develop.

2) После завершения разработки фичевая ветка сливается обратно в ветку Develop.

3) Периодически, когда необходимо выпустить новую версию, создается ветка Release, на которой проводятся завершающие работы и исправления.

4) Ветка Release сливается как в ветку Master, так и в ветку Develop.

5) В случае непредвиденных проблем или ошибок на ветке Master создается ветка Hotfix для внесения исправлений в продукт.

6) Ветка Hotfix также сливается как в ветку Master, так и в ветку Develop.

Недостатки модели `git-flow` включают следующее: сложность, медленный релиз, отсутствие гибкости, частые конфликты слияния.

13. В GitHub Desktop есть несколько инструментов, которые вы можете использовать для работы с ветками:

- Переключение между ветками;
- Создание новой ветки;
- Слияние веток;
- Удаление ветки;
- Отслеживание удаленных веток;