

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 №1.3
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Медяник Даниил Владимирович
1 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

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

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Основы ветвления Git.

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

Ход работы:

Задание 1. Создал общедоступный репозиторий в котором будет использована лицензия MIT. Клонировал репозиторий.

```
C:\Users\Gaming-PC>git clone https://github.com/DaniilMedy/laba-1.3.git
Cloning into 'laba-1.3'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
```

Рисунок 1. Новый репозиторий

Задание 2. Добавил 3 текстовых файла.







Имя	Дата изменения	Тип	Размер
 .gitignore	06.06.2023 20:13	Текстовый докум...	1 КБ
 1.txt	06.06.2023 20:14	Текстовый докум...	0 КБ
 2.txt	06.06.2023 20:14	Текстовый докум...	0 КБ
 3.txt	06.06.2023 20:14	Текстовый докум...	0 КБ
 LICENSE	06.06.2023 20:13	Файл	2 КБ
 README	06.06.2023 20:13	Исходный файл ...	1 КБ

Рисунок 2. Создание текстовых файлов

Задание 3. Проиндексировал первый файл и сделал коммит. Проиндексировал второй и третий файлы.

```
C:\Users\Gaming-PC\laba-1.3>git add 1.txt
C:\Users\Gaming-PC\laba-1.3>git commit -m "add 1.txt"
[main d30603f] add 1.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
C:\Users\Gaming-PC\laba-1.3>git add 2.txt
C:\Users\Gaming-PC\laba-1.3>git commit -m "add 2.txt"
[main debcbb1] add 2.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 2.txt
C:\Users\Gaming-PC\laba-1.3>git add 3.txt
C:\Users\Gaming-PC\laba-1.3>git commit -m "add 3.txt"
[main ad3d97f] add 3.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 3.txt
```

Рисунок 3. Добавление изменений

1. Перезаписал уже сделанный коммит.

```
C:\Users\Gaming-PC\laba-1.3>git commit --amend -m "add 2.txt and 3.txt"
[main ec1f45d] add 2.txt and 3.txt
Date: Tue Jun 6 20:19:09 2023 +0300
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 3.txt
```

Рисунок 4. Редактирование существующего коммита

2. Создал новую ветку my_first_branch

```
C:\Users\Gaming-PC\laba-1.3>git branch my_first_branch
```

Рисунок 5. Новая ветка

3. Перейти на ветку и создать новый файл in_branch.txt.

The image shows a Windows File Explorer window on the left, displaying the contents of a directory: .gitignore, 1, 2, 3, in_branch, LICENSE, and README. The 'in_branch' folder is selected. On the right, a terminal window shows the following commands and output:

```
[main ec1f45d] add 2.txt and 3.txt
Date: Tue Jun 6 20:19:09 2023 +0300
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 3.txt

C:\Users\Gaming-PC\laba-1.3>git branch my_first_branch

C:\Users\Gaming-PC\laba-1.3> git add in_branch.txt

C:\Users\Gaming-PC\laba-1.3>git commit -m "in_branch"
[main f59efc1] in_branch
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

C:\Users\Gaming-PC\laba-1.3>
```

Рисунок 6. Создание файла и фиксация

4. Перешел вновь на новую ветку main.

```
C:\Users\Gaming-PC\laba-1.3>git checkout main
Already on 'main'
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)

C:\Users\Gaming-PC\laba-1.3>git branch
* main
  my_first_branch
```

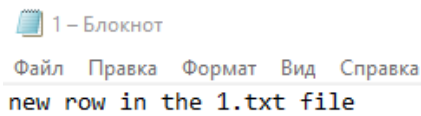
Рисунок 7. Переход на главную ветвь

5. Создал и сразу перешел на ветку new_branch.

```
C:\Users\Gaming-PC\laba-1.3>git checkout -b new_branch
Switched to a new branch 'new_branch'
```

Рисунок 8. Создание и сразу переход на ветку

6. Сделал изменения в файле 1.txt, добавил строчку “new row in the 1.txt file”.



```
C:\Users\Gaming-PC\laba-1.3>git branch
* main
  my_first_branch

C:\Users\Gaming-PC\laba-1.3>git checkout -b new_branch
Switched to a new branch 'new_branch'

C:\Users\Gaming-PC\laba-1.3>git add 1.txt

C:\Users\Gaming-PC\laba-1.3>git commit -m "1.txt"
[new_branch 67008b6] 1.txt
1 file changed, 1 insertion(+)

C:\Users\Gaming-PC\laba-1.3>
```

Рисунок 9. Изменение в 1.txt фиксация этих изменений

7. Перешел на ветку main и слил ветки main и my_first_branch, после чего слил ветки main и new_branch

```
C:\Users\Gaming-PC\laba-1.3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)

C:\Users\Gaming-PC\laba-1.3>git branch
* main
  my_first_branch
  new_branch

C:\Users\Gaming-PC\laba-1.3>git merge new_branch
Updating f59efc1..67008b6
Fast-forward
 1.txt | 1 +
1 file changed, 1 insertion(+)
```

Рисунок 10. Слияние веток

8. Удалил ветки my_first_branch и new_branch.

```
C:\Users\Gaming-PC\laba-1.3>git branch -d my_first_branch
Deleted branch my_first_branch (was ec1f45d).

C:\Users\Gaming-PC\laba-1.3>git branch -d new_branch
Deleted branch new_branch (was 67008b6).
```

Рисунок 11. Удаление веток

9. Создал ветки branch_1 и branch_2.

```
C:\Users\Gaming-PC\laba-1.3>git branch branch_1

C:\Users\Gaming-PC\laba-1.3>git branch branch_2

C:\Users\Gaming-PC\laba-1.3>git branch
branch_1
branch_2
* main
```

Рисунок 12. Создание новых веток

10. Перешел на ветку `branch_1` и изменил файл `1.txt`, удалил все содержимое и добавил текст “fix in the 1.txt”, изменил файл `3.txt`, удалил все содержимое и добавил текст “fix in the 3.txt”, закоммитил изменения.

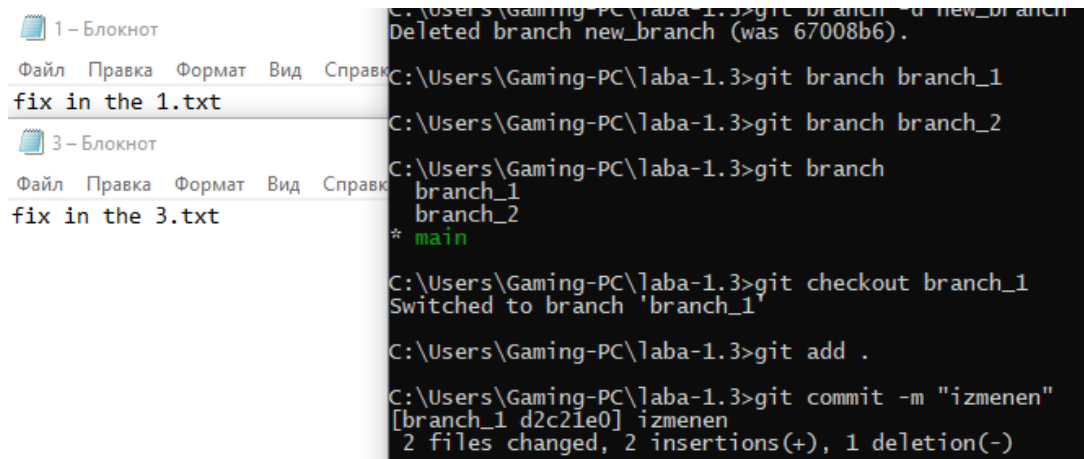


Рисунок 13. Изменения в первой ветке и во второй ветка

11. Слил изменения ветки `branch_2` в ветку `branch_1`.

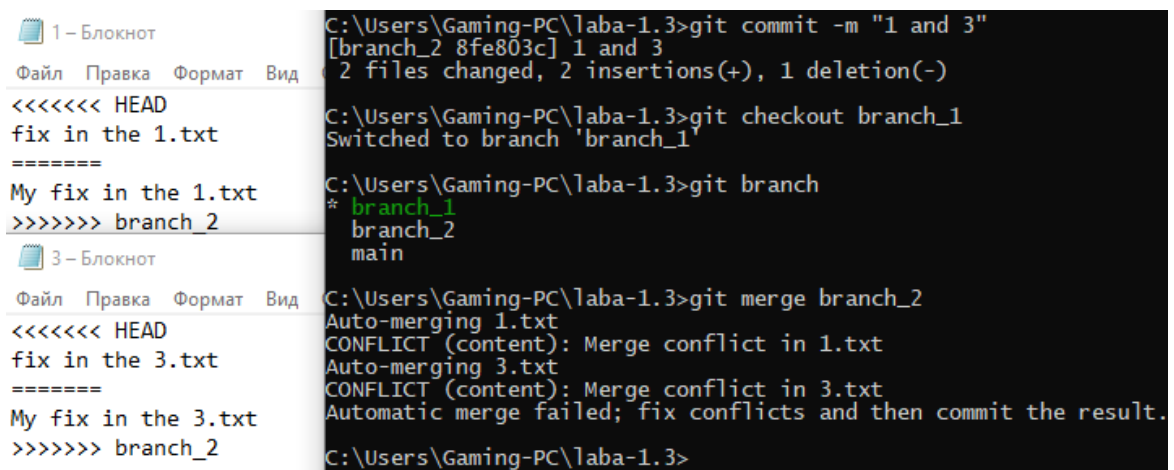


Рисунок 14. Слияние веток

12. Решил конфликт файла `1.txt` в ручном режиме.

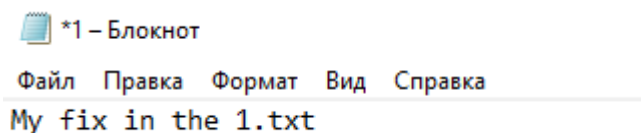


Рисунок 15. Решение конфликта вручную.

13. Необходимо изменить еще 3 текстовый файл, но уже с помощью инструмента.

```

C:\Users\Gaming-PC\laba-1.3>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

C:\Users\Gaming-PC\laba-1.3>git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
tortoisemerge emerge vimdiff nvimdiff
Merging:
3.txt

Normal merge conflict for '3.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
C:\Users\Gaming-PC\laba-1.3>

```

Рисунок 16. Решение конфликта с помощью инструмента

14. Отправил branch_1 на удаленный сервер с помощью команды: `git push origin branch_1`.

branch_1
 2 branches
 0 tags
 Go to file
Add file
Code

This branch is 8 commits ahead of main.
 Contribute

	DaniilMedy 3.txt	a6696a0 now	10 commits
	.gitignore	Initial commit	yesterday
	1.txt	3.txt	now
	2.txt	add 2.txt	1 hour ago
	3.txt	3.txt	now
	LICENSE	Initial commit	yesterday
	README.md	add	yesterday
	in_branch.txt	in_branch	49 minutes ago

Рисунок 17. Отправление ветки на удаленный сервер

15. Создал средствами GitHub удаленную ветку branch_3.

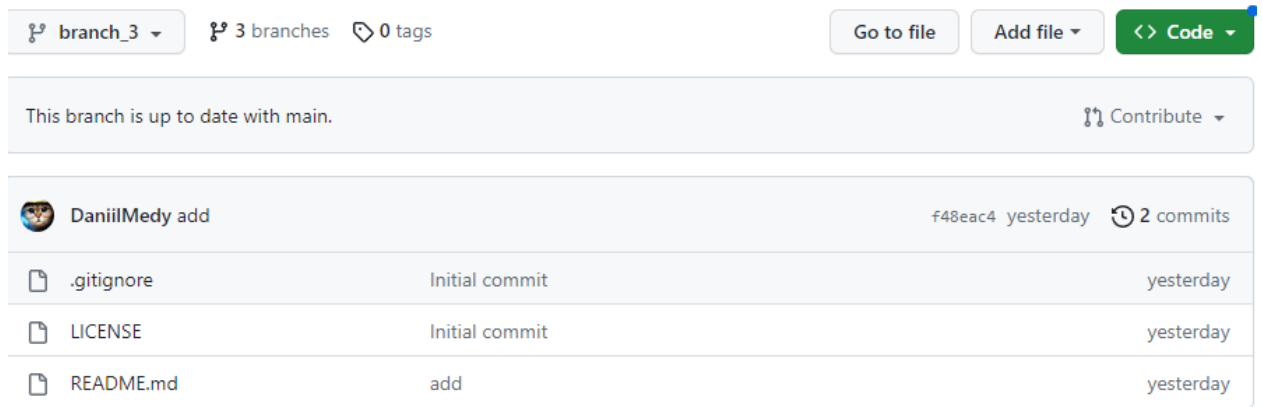


Рисунок 18. Создание ветки на GitHub

16. Создал в локальном репозитории ветку отслеживания удаленной ветки branch_3.

```
C:\Users\Gaming-PC\laba-1.3>git fetch origin
From https://github.com/DaniilMedy/laba-1.3
* [new branch]      branch_3 -> origin/branch_3

C:\Users\Gaming-PC\laba-1.3>git checkout -b branch_3 origin/branch_3
Switched to a new branch 'branch_3'
branch 'branch_3' set up to track 'origin/branch_3'.
```

Рисунок 19. Создание ветки отслеживания

17. Перешел на ветку branch_3 и добавил файл файл 2.txt строку "the final fantasy in the 4.txt file".

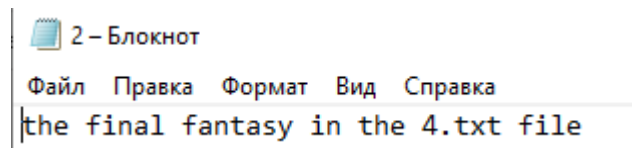


Рисунок 20. Изменение файла во второй ветке

18. Выполнил перемещение ветки main на ветку branch_2.

```
C:\Users\Gaming-PC\laba-1.3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 5 commits.
(use "git push" to publish your local commits)

C:\Users\Gaming-PC\laba-1.3>git rebase branch_2
Successfully rebased and updated refs/heads/main.

C:\Users\Gaming-PC\laba-1.3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\Gaming-PC\laba-1.3>git merge main
Already up to date.
```

Рисунок 21. Перемещение веток

19. Отправил изменения веток master и branch_2.

```
C:\Users\Gaming-PC\laba-1.3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 6 commits.
  (use "git push" to publish your local commits)

C:\Users\Gaming-PC\laba-1.3>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/DaniilMedy/laba-1.3.git
   f48eac4..8fe803c  main -> main

C:\Users\Gaming-PC\laba-1.3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\Gaming-PC\laba-1.3>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/DaniilMedy/laba-1.3/pull/new/branch_2
remote:
To https://github.com/DaniilMedy/laba-1.3.git
 * [new branch]      branch_2 -> branch_2
```

Рисунок 22. Отправка изменения

Ответы на контрольные вопросы:

1. Что такое ветка?

Почти каждая система контроля версий (СКВ) в какой-то форме поддерживает ветвление. Используя ветвление, Вы отклоняетесь от основной линии разработки и продолжаете работу независимо от неё, не вмешиваясь в основную линию.

2. Что такое HEAD?

HEAD в Git-это указатель на текущую ссылку ветви, которая, в свою очередь, является указателем на последний сделанный вами коммит или последний коммит, который был извлечен из вашего рабочего каталога.

3. Способы создания веток.

Создать ветку можно с помощью двух команд. Команда, которая просто создает ветку: `git branch "name_branch"`.

Команда, которая создает ветку и сразу же к ней переходит: `git checkout -b "name_branch"`.

4. Как узнать текущую ветку?

Текущую ветку можно узнать с помощью команды: `git branch`.

5. Как переключаться между ветками?

Между ветками можно переключаться с помощью команды: `git checkout "name_branch"`.

6. Что такое удаленная ветка?

Удаленные ветки - это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые нельзя перемещать.

7. Что такое ветка отслеживания?

Отслеживаемые ветки — это локальные ветки, которые напрямую связаны с удалённой веткой.

Если, находясь на отслеживаемой ветке, вы наберёте `git push`, Git уже будет знать, на какой сервер и в какую ветку отправлять изменения.

8. Как создать ветку отслеживания?

Ветку отслеживания можно создать с помощью команды: `git checkout --track origin/<name_branch>`.

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

Отправить изменения из локальной ветки в удаленную можно с помощью команды: `git push origin <name_branch>`.

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`.

11. Как удалить локальную и удаленную ветки?

Для удаление локальной ветки используется команда: `git branch -d <name_branch>`

Для удаления удаленной ветки используется команда: `git push --delete origin/<name_branch>`

12. Какие основные типы веток присутствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?

Существуют следующие типы ветвей:

- 1) ветви функциональностей;
- 2) ветви релизов;
- 3) ветви исправлений.

Ветви функциональностей (feature branches), также называемые иногда тематическими ветвями (topic branches), используются для разработки новых функций, которые должны появиться в текущем или будущем релизах.

Ветви релизов (release branches) используются для подготовки к выпуску новых версий продукта. Они позволяют расставить финальные точки над *i* перед выпуском новой версии.

Ветви для исправлений (hotfix branches) весьма похожи на ветви релизов (release branches), так как они тоже используются для подготовки новых выпусков продукта, разве лишь незапланированных.

Недостатки git flow: авторам приходится использовать ветку develop вместо master, поскольку master зарезервирован для кода.

Вторая проблема процесса git flow – сложности, возникающие из-за веток для патчей и для релиза. Подобная структура может подойти некоторым организациям, но для абсолютного большинства она просто убийственно излишня.

Вывод: исследовал базовые возможности по работе с локальными и удаленными ветками Git.