

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 РАБОТЕ №1.3
дисциплины «Основы кроссплатформенного программирования»
Вариант 3

Выполнил:
Баканов Артем Вадимович
1 курс, группа ИТС-6-0-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: основы ветвления Git.

Цель работы: исследовать базовые возможности по работе с локальными и удаленными ветками Git.

Ссылка: <https://github.com/temacteklyannayapuwka/cross-platform-programming-v.3>

Порядок выполнения работы:

1. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT.

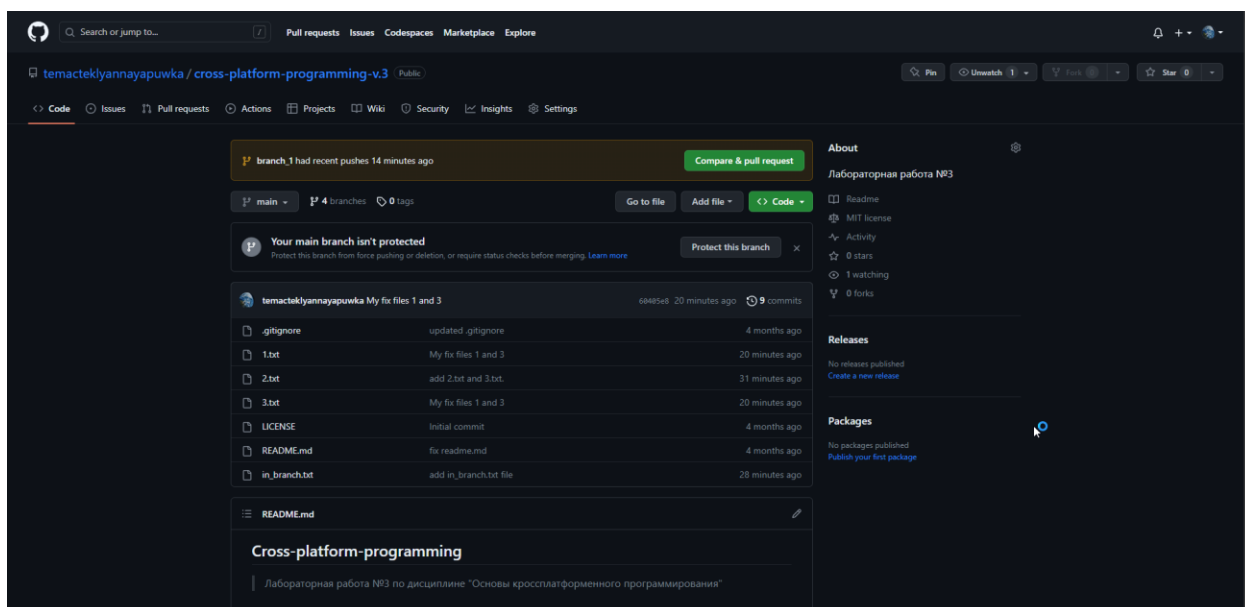


Рис. 1. Новый репозиторий.

2. Добавил 3 новых текстовых файла.

3.txt	6/7/2023 5:34 PM	TXT File	1 KB
2.txt	6/7/2023 5:33 PM	TXT File	0 KB
1.txt	6/7/2023 5:34 PM	TXT File	1 KB

Рис. 2. Новые текстовые файлы.

3. Проиндексировал первый файл и сделать коммит.

```
PS D:\Crosslabs\cross-platform-programming-v.3> git add 1.txt
PS D:\Crosslabs\cross-platform-programming-v.3> git commit -m "add 1.txt file"
[main ce353a2] add 1.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
```

Рис. 3. Добавление изменений и их фиксация.

4. Перезаписал уже сделанный коммит.

```
PS D:\Crosslabs\cross-platform-programming-v.3> git commit --amend
[main efe4665] add 1.txt file
Date: Wed Jun 7 17:05:54 2023 +0300
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
create mode 100644 2.txt
create mode 100644 3.txt
```

Рис. 4. Редактирование существующего коммита.

5. Создание новой ветки и переход на нее и создать новый файл in_branch.txt.

```
PS D:\Crosslabs\cross-platform-programming-v.3> git branch my_first_branch
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout my_first_branch
Switched to branch 'my_first_branch'
PS D:\Crosslabs\cross-platform-programming-v.3> git add .
PS D:\Crosslabs\cross-platform-programming-v.3> git commit -m "add in_branch.txt file"
[my_first_branch e6d9bfc] add in_branch.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt
```

Рис. 5. Создание файла и фиксация.

9. Перешел вновь на новую ветку main и создал и сразу перешел на ветку new_branch.

```
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout -b new_branch
Switched to a new branch 'new_branch'
```

Рис. 9. Создание и сразу переход на ветку.

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

```
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout -b new_branch
Switched to a new branch 'new_branch'
PS D:\Crosslabs\cross-platform-programming-v.3> git add 1.txt
PS D:\Crosslabs\cross-platform-programming-v.3> git commit -m "new row in the 1.txt"
[new_branch a591d4f] new row in the 1.txt
1 file changed, 1 insertion(+)
```

Рис. 10. Изменение в 1.txt фиксация этих изменений.

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

```
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS D:\Crosslabs\cross-platform-programming-v.3> git merge my_first_branch
Updating fae59a3..e6d9bfc
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt
PS D:\Crosslabs\cross-platform-programming-v.3> git merge new_branch
Merge made by the 'ort' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)
```

Рис. 11. Слияние веток.

12. Удалил ветки my_first_branch и new_branch.

```
PS D:\Crosslabs\cross-platform-programming-v.3> git branch -d my_first_branch
Deleted branch my_first_branch (was e6d9bfc).
PS D:\Crosslabs\cross-platform-programming-v.3> git branch -d new_branch
Deleted branch new_branch (was a591d4f).
```

Рис. 12. Удаление веток

13. Создал ветки branch_1 и branch_2.

```
PS D:\Crosslabs\cross-platform-programming-v.3> git branch branch_1
PS D:\Crosslabs\cross-platform-programming-v.3> git branch branch_2
```

Рис. 13. Создание новых веток.

14. 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.

```
PS D:\Crosslabs\cross-platform-programming-v.3> git add .
PS D:\Crosslabs\cross-platform-programming-v.3> git commit -m "fix files 1 and 3"
[branch_1 f93497d] fix files 1 and 3
 2 files changed, 2 insertions(+), 1 deletion(-)
```

Рис. 14. Изменения в первой ветке и во второй ветка и фиксация изменений.

15. Перешел на ветку branch_2 и также изменил файл 1.txt, удалил все содержимое и добавил текст “My fix in the 1.txt”, изменил файл 3.txt, удалил все содержимое и добавил текст “My fix in the 3.txt”, закоммитил изменения.

```
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout branch_2
Switched to branch 'branch_2'
PS D:\Crosslabs\cross-platform-programming-v.3> git add .
PS D:\Crosslabs\cross-platform-programming-v.3> git commit -m "My fix files 1 and 3"
[branch_2 60405e8] My fix files 1 and 3
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рис. 15. Переход на вторую ветку.

16. Слил изменения ветки branch_2 в ветку branch_1.

```
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout branch_1
Switched to branch 'branch_1'
PS D:\Crosslabs\cross-platform-programming-v.3> 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.
```

Рис. 16. Слияние веток.

17. Решил конфликт файла 1.txt и 2.txt

```
PS D:\Crosslabs\cross-platform-programming-v.3> git add .
PS D:\Crosslabs\cross-platform-programming-v.3> git merge branch_2
fatal: You have not concluded your merge (MERGE_HEAD exists).
Please, commit your changes before you merge.
PS D:\Crosslabs\cross-platform-programming-v.3> git commit -m "fix conflict"
[branch_1 c8d8b0e] fix conflict
PS D:\Crosslabs\cross-platform-programming-v.3> git merge branch_2
Already up to date.
PS D:\Crosslabs\cross-platform-programming-v.3> git status
On branch branch_1
nothing to commit, working tree clean
```

Рис. 17. Решение конфликта

18. Отправил branch_1 на удаленный git push origin branch_1.

```
PS D:\Crosslabs\cross-platform-programming-v.3> git push --set-upstream origin branch_1
Enumerating objects: 22, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 4 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (19/19), 1.65 KiB | 422.00 KiB/s, done.
Total 19 (delta 8), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (8/8), completed with 1 local object.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:   https://github.com/temacteklyannayapuwka/cross-platform-programming-v.3/pull/new/branch_1
remote:
To https://github.com/temacteklyannayapuwka/cross-platform-programming-v.3.git
 * [new branch]      branch_1 -> branch_1
branch 'branch_1' set up to track 'origin/branch_1'.
```

Рис. 18. Отправление ветки на удаленный сервер.

18. Создал средствами GitHub удаленную ветку branch_3.

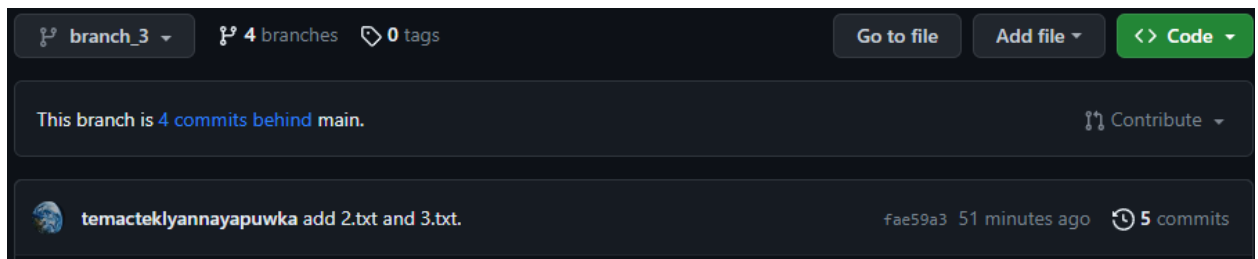


Рис. 18. Создание ветки на GitHub.

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

```
PS D:\Crosslabs\cross-platform-programming-v.3> git fetch origin
From https://github.com/temacteklyannayapuwka/cross-platform-programming-v.3
 * [new branch]      branch_3 -> origin/branch_3
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout -b branch_3 origin/branch_3
Switched to a new branch 'branch_3'
```

Рис. 19. Создание ветки отслеживания.

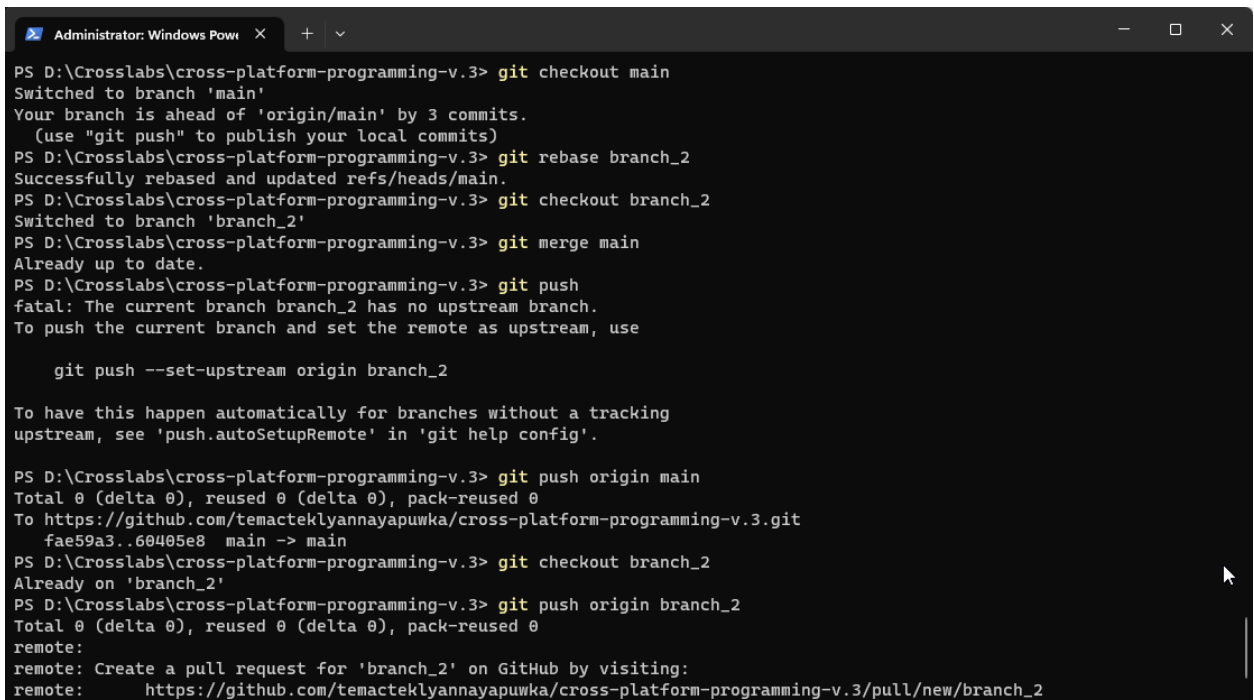
20. Перешел на ветку branch_3 и добавил файл файл 2.txt строку "the final fantasy in the 4.txt file".

21. Выполнил перемещение ветки main на ветку branch_2.

```
PS D:\Crosslabs\cross-platform-programming-v.3> git add .
PS D:\Crosslabs\cross-platform-programming-v.3> git commit -m "the final fantasy in the 4.txt"
[branch_3 ebe0fc5] the final fantasy in the 4.txt
1 file changed, 1 insertion(+)
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 3 commits.
(use "git push" to publish your local commits)
PS D:\Crosslabs\cross-platform-programming-v.3> git rebase branch_2
Successfully rebased and updated refs/heads/main.
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout branch_2
Switched to branch 'branch_2'
PS D:\Crosslabs\cross-platform-programming-v.3> git merge main
Already up to date.
```

Рис. 25. Перемещение веток.

22. Отправил изменения веток master и branch_2 на удаленный сервер.



```
Administrator: Windows Powe... X + v
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 3 commits.
(use "git push" to publish your local commits)
PS D:\Crosslabs\cross-platform-programming-v.3> git rebase branch_2
Successfully rebased and updated refs/heads/main.
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout branch_2
Switched to branch 'branch_2'
PS D:\Crosslabs\cross-platform-programming-v.3> git merge main
Already up to date.
PS D:\Crosslabs\cross-platform-programming-v.3> git push
fatal: The current branch branch_2 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin branch_2

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
PS D:\Crosslabs\cross-platform-programming-v.3> git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/temacteklyannayapuwka/cross-platform-programming-v.3.git
fae59a3..60405e8  main -> main
PS D:\Crosslabs\cross-platform-programming-v.3> git checkout branch_2
Already on 'branch_2'
PS D:\Crosslabs\cross-platform-programming-v.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/temacteklyannayapuwka/cross-platform-programming-v.3/pull/new/branch_2
```

Рис. 22. Отправка изменения веток на GitHub.

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 определенное 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/`.

9. Как отправить изменения из локальной ветки в удаленную ветку? Отправить изменения из локальной ветки в удаленную можно с помощью команды: `git push origin` .

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

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.