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 «Программирование на python»

Выполнил:
Кожуховский Виктор Андреевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств вычислительной
техники и автоматизированных систем
», очная форма обучения

_____
(подпись)

Руководитель практики:
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.

Методика и порядок выполнения работы

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

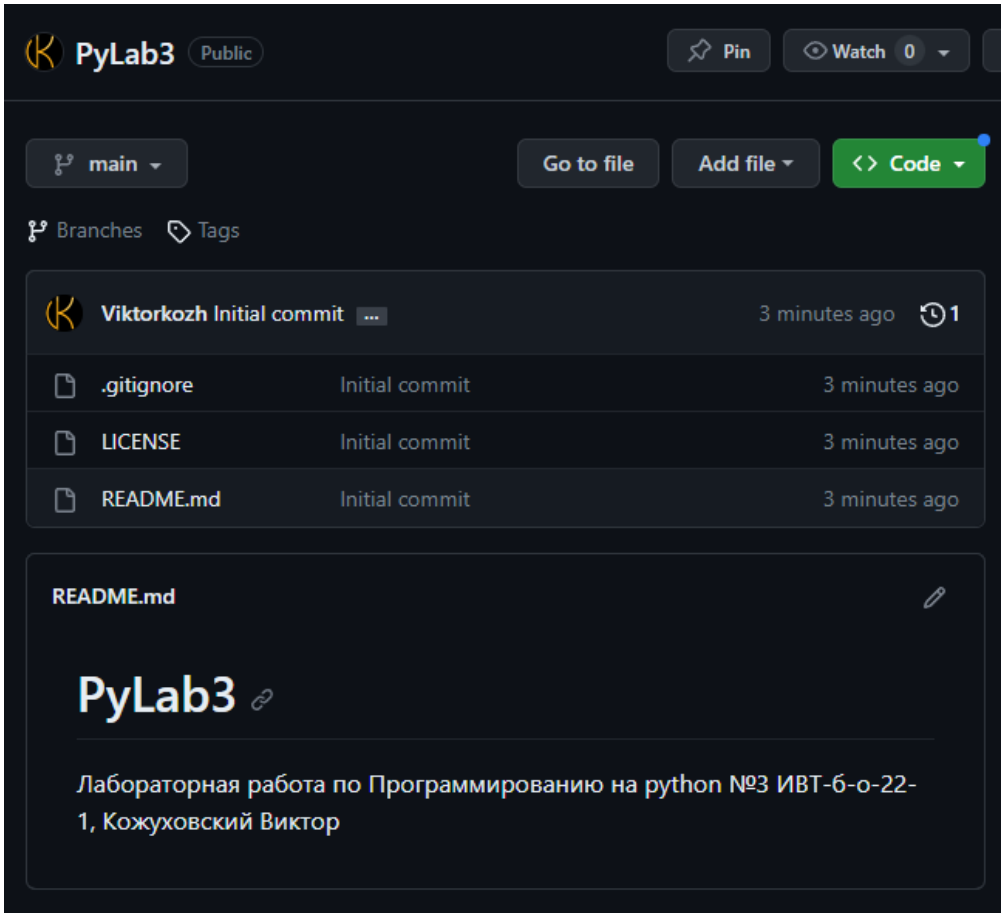

Рисунок 1. Созданный репозиторий

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

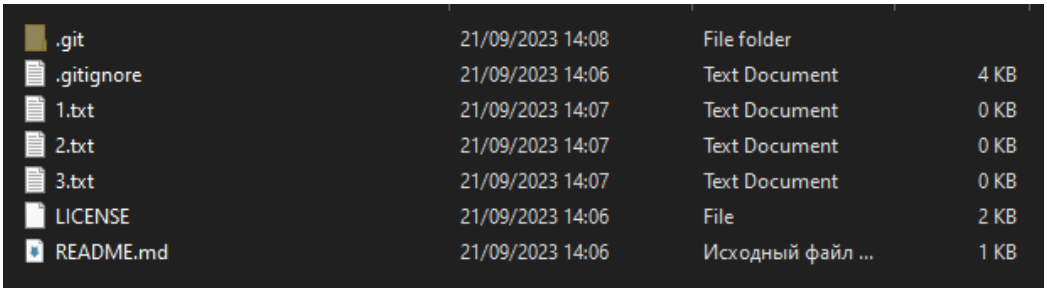

Рисунок 2. Созданные текстовые файлы в папке репозитория

3. Проиндексировал первый файл и сделал коммит с комментарием "add 1.txt file", проиндексировал второй и третий файлы, перезаписал уже сделанный коммит с новым комментарием "add 2.txt and 3.txt."

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git add 1.txt
fatal: pathspec '1.txt' did not match any files
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git add 1.txt
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git commit -m "Added 1.txt file"
[main 453af0d] Added 1.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git add --all
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git commit -m "Added 2.txt and 3.txt file"
[main 1e1aef4] Added 2.txt and 3.txt file
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2.txt
 create mode 100644 3.txt
```

Рисунок 3. Перезаписанный коммит

4. Создал новую ветку my_first_branch, перешел на ветку и создал новый файл in_branch.txt, закоммитил изменения.

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git branch my_first_branch
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git checkout my_first_branch
Switched to branch 'my_first_branch'
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git add in_branch.txt
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git commit -m "Added in_branch.txt"
[my_first_branch cbec5de] Added in_branch.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git push
fatal: The current branch my_first_branch has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin my_first_branch

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git push --set-upstream origin my_first_branch
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 730 bytes | 730.00 KiB/s, done.
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'my_first_branch' on GitHub by visiting:
remote:      https://github.com/Viktorkozh/PyLab3/pull/new/my_first_branch
remote:
To https://github.com/Viktorkozh/PyLab3
 * [new branch]      my_first_branch -> my_first_branch
branch 'my_first_branch' set up to track 'origin/my_first_branch'.
```

Рисунок 4. Коммит запушен

5. Вернулся на ветку origin.

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git checkout origin
Note: switching to 'origin'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 02b9c24 Initial commit
PS C:\Us
```

Рисунок 5. Возврат на ветку origin

6. Создал и сразу перешел на ветку new_branch.

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git branch new_branch
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git chechout new_branch
git: 'chechout' is not a git command. See 'git --help'.

The most similar command is
        checkout
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git checkout new_branch
Switched to branch 'new_branch'
```

Рисунок 6. Создание новой ветки и переход на нее

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

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git add --all
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git commit -m "new row in the 1.txt file"
[my_first_branch 147bf39] new row in the 1.txt file
 1 file changed, 1 insertion(+)
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 306 bytes | 306.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Viktorkozh/PyLab3
   cbec5de..147bf39  my_first_branch -> my_first_branch
```

Рисунок 7. Пуш на новую ветку

8. Перешел на ветку origin и слил ветки origin и my_first_branch, после чего слил ветки master и new_branch.

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git merge my_first_branch origin
Updating 02b9c24..147bf39
Fast-forward
 1.txt         | 1 +
 2.txt         | 0
 3.txt         | 0
 in_branch.txt | 0
 4 files changed, 1 insertion(+)
 create mode 100644 1.txt
 create mode 100644 2.txt
 create mode 100644 3.txt
 create mode 100644 in_branch.txt
```

Рисунок 8. Мердж веток

9. Удалил ветки my_first_branch и new_branch.

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git branch -d my_first_branch
Deleted branch my_first_branch (was 147bf39).
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git branch -d new_branch
Deleted branch new_branch (was 02b9c24).
```

Рисунок 9. Удаление веток

10. Создал ветки branch_1 и branch_2.

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git branch branch_1
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git branch branch_2
```

Рисунок 10. Создание веток branch_1 и branch_2

11. 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.

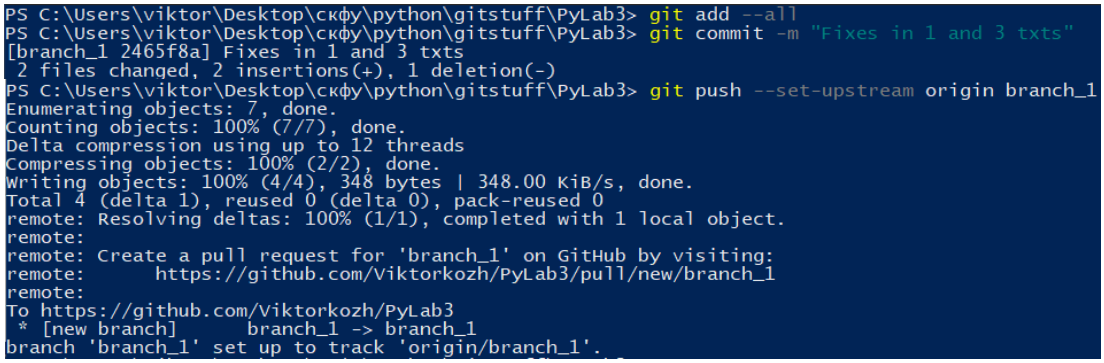

Рисунок 11. Пуш изменений на ветку branch_1

12. Перешел на ветку branch_2 и также изменить 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.

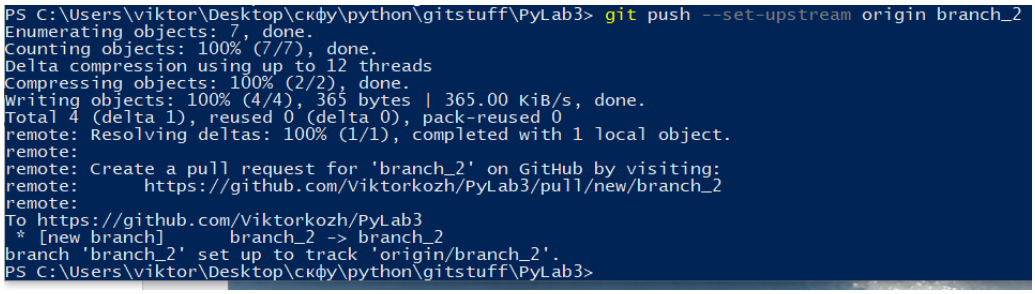

Рисунок 12. Коммит с изменениями запушен

13. Слил изменения ветки branch_2 в ветку branch_1.

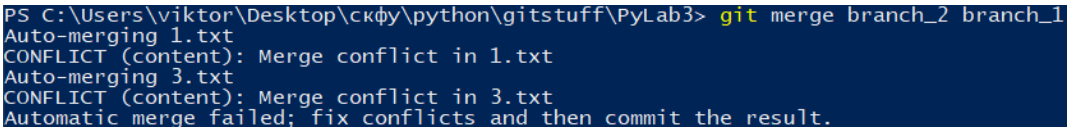

Рисунок 13. Попытка слива изменений

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

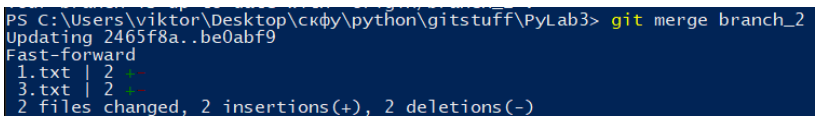

Рисунок 14. Решил конфликт и выполнил git merge

15. Отправил ветку branch_1 на GitHub.

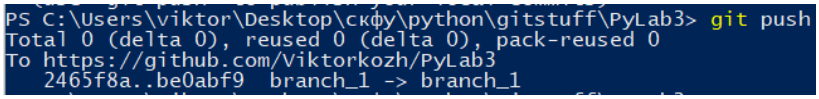

Рисунок 15. Ветка branch_1 отправлена

16. Создал средствами GitHub удаленную ветку branch_3.

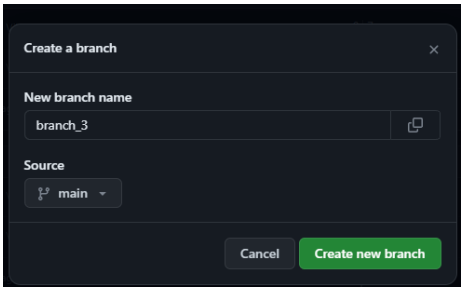

Рисунок 16. Создание ветки branch_3 средствами GitHub

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

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git checkout branch_3
error: pathspec 'branch_3' did not match any file(s) known to git
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git pull
From https://github.com/Viktorkozh/PyLab3
 * [new branch]      branch_3   -> origin/branch_3
Already up to date.
```

Рисунок 17. Выполнение git pull

18. Перешел на ветку branch_3 и добавил файл файл 2.txt строку "the final fantasy in the 4.txt file". Выполнил перемещение ветки origin на ветку branch_2.

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git checkout branch_2
Switched to branch 'branch_2'
M       2.txt
Your branch is up to date with 'origin/branch_2'.
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git pull origin branch_2
From https://github.com/Viktorkozh/PyLab3
 * branch            branch_2   -> FETCH_HEAD
Already up to date.
```

Рисунок 18. Перемещение ветки оригин на ветку branch_2

19. Отправил изменения веток origin и branch_2 на GitHub.

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git checkout branch_2
Switched to branch 'branch_2'
M       2.txt
Your branch is up to date with 'origin/branch_2'.
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git pull origin branch_2
From https://github.com/Viktorkozh/PyLab3
 * branch            branch_2   -> FETCH_HEAD
Already up to date.
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git push
Everything up-to-date
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git add --all
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git commit -m "s"
[branch_2 a7dfe0b] s
 1 file changed, 1 insertion(+)
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 285 bytes | 285.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Viktorkozh/PyLab3
   be0abf9..a7dfe0b  branch_2 -> branch_2
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> 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)
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git add --all
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git commit -m "s"
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab3> git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Viktorkozh/PyLab3
   02b9c24..1e1aef4  main -> main
```

Рисунок 19. Пуши origin и branch_2

Вопросы для защиты работы

1. Что такое ветка?

Ветка в Git — это простой перемещаемый указатель на один из коммитов.

2. Что такое HEAD?

HEAD – это указатель, задача которого ссылаться на определенный коммит в репозитории.

HEAD – это указатель на коммит в вашем репозитории, который станет родителем следующего коммита.

HEAD указывает на коммит, относительного которого будет создана рабочая копия во время операции checkout.

3. Способы создания веток.

Чтобы создать новую ветку, необходимо использовать команду git branch.

Чтобы создать ветку и сразу переключиться на нее, можно использовать команду git checkout -b.

4. Как узнать текущую ветку?

Увидеть текущую ветку можно при помощи простой команды git log, которая покажет куда указывают указатели веток. Эта опция называется -decorate. HEAD будет стоять рядом с текущей веткой. Также можно использовать git branch -v, рядом с текущей веткой будет значек *.

5. Как переключаться между ветками?

Для переключения на существующую ветку выполните команду git checkout.

6. Что такое удаленная ветка?

Удалённые ссылки — это ссылки (указатели) в ваших удалённых репозиториях, включая ветки, теги и так далее. Полный список удалённых ссылок можно получить с помощью команды git ls-remote <remote> или команды git remote show <remote> для получения удалённых веток и дополнительной информации.

7. Что такое ветка отслеживания?

Ветки слежения — это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые нельзя перемещать; Git перемещает их автоматически при любой коммуникации с удаленным репозиторием, чтобы гарантировать точное соответствие с ним.

8. Как создать ветку отслеживания?

При клонировании репозитория, как правило, автоматически создаётся ветка master, которая следит за origin/master. Однако, при желании можно настроить отслеживание и других веток — следить за ветками на других серверах или отключить слежение за веткой master. Сделать это можно с помощью команды git checkout -b <branch> <remote>/<branch>. Это часто используемая команда, поэтому Git предоставляет сокращённую форму записи в виде флага --track.

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

Когда вы хотите поделиться веткой, вам необходимо отправить её на удалённый сервер, где у вас есть права на запись. Ваши локальные ветки автоматически не синхронизируются с удалёнными при отправке — вам нужно явно указать те ветки, которые вы хотите отправить.

Таким образом, вы можете использовать свои личные ветки для работы, которую не хотите показывать, а отправлять только те тематические ветки, над которыми вы хотите работать с кем-то совместно. Отправка осуществляется командой git push <remote> <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 .

Если у вас настроена ветка слежения, 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 локальной ветки выполните команду git branch с параметром -d.

Вы можете удалить ветку на удалённом сервере используя параметр --delete для команды git push .

12. Изучить модель ветвления git-flow (использовать материалы статей https://www.atlassian.com/ru/git/tutorials/comparing-workflows/gitflow-workflow, https://hab1r.com/ru/post/106912/). 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 ветвления git-flow, основные типы веток это:

master - главная ветка проекта, которая содержит только стабильный код и используется для создания релизов.

develop - ветка, в которой ведется основная разработка проекта. Она содержит все изменения, которые были сделаны разработчиками.

feature - ветки для добавления новой функциональности в проект. Они ветвятся от ветки develop, и после завершения работы ветки объединяются с develop веткой.

release - ветки для подготовки новой версии проекта. Они ветвятся от ветки develop, содержат минимальный набор изменений и используются для подготовки релиза. После тестирования и отладки, ветка объединяется с веткой master и develop.

hotfix - ветки для исправления критических ошибок на производственной версии проекта. Они ветвятся от ветки master, и после исправления ошибок объединяются с master и develop ветками.

Работа с ветками в модели git-flow организована следующим образом:

Начало разработки новой функциональности начинается с ветвления от ветки develop ветки feature, на которой работает разработчик.

После завершения работы, все изменения ветки feature тестируются, затем вливаются обратно в ветку develop.

В момент подготовки новой версии программного продукта ветка release создается из develop, и она используется для проведения основных тестов наиболее важных функций.

Ветка hotfix создается из ветки master, если в производственной версии обнаружена критическая ошибка, и на этой ветке выполняется исправление.

Недостатки git-flow включают в себя:

Сложность и необходимость управления множеством веток, что может быть трудным для маленьких команд.

Не слишком хорошо подходит для быстрой разработки и быстрой реализации необходимых исправлений или функций в связи с наличием большого количества ветвлений.

Накладывает значительный набор процедур и правил для разработки и управления релизами.

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.

Codeberg.org предоставляет доступ к репозиториям Git и предоставляет возможности для работы с ветками, аналогично большинству других хостингов репозиториев Git. Вот некоторые из основных инструментов и команд, которые можно использовать для работы с ветками на Codeberg.org:

1. `git branch`: Эта команда позволяет просматривать список существующих веток в вашем локальном репозитории.

2. `git checkout`: С помощью этой команды можно переключаться между существующими ветками в локальном репозитории.

3. `git checkout -b new-branch-name`: Эта команда создает новую ветку с указанным именем и переключается на нее.

4. `git push origin branch-name`: Для отправки изменений из вашей локальной ветки в ветку на Codeberg.org, можно использовать эту команду, где `branch-name` - это имя ветки.

5. Codeberg.org также предоставляет веб-интерфейс для работы с репозиториями, который включает в себя возможность просмотра, создания и удаления веток.