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 программной инженерии»

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

_____
(подпись)

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

Результаты выполнения

```
itssyoma@MacBook-Air-Sema Основы программной инженерии % git clone https://githu
b.com/itssyoma/megarepo_13.git
Cloning into 'megarepo_13'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
itssyoma@MacBook-Air-Sema Основы программной инженерии %
```

Рисунок 1 – Клонирование нового репозитория.

```
itssyoma@MacBook-Air-Sema megarepo_13 % touch 1.txt
itssyoma@MacBook-Air-Sema megarepo_13 % touch 2.txt
itssyoma@MacBook-Air-Sema megarepo_13 % touch 3.txt
itssyoma@MacBook-Air-Sema megarepo_13 % git add 1.txt
itssyoma@MacBook-Air-Sema megarepo_13 % git commit -m "add 1.txt file"
[main e640c54] add 1.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt
```

Рисунок 2 – Создание текстовых файлов, добавление 1.txt в индекс и его коммит.

```
itssyoma@Air-Sema megarepo_13 % git commit --amend
[main 7ff09cf] add 2.txt and 3.txt
 Date: Wed Oct 11 09:58:47 2023 +0300
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt
 create mode 100644 2.txt
 create mode 100644 3.txt
itssyoma@Air-Sema megarepo_13 % git log
commit 7ff09cf73eb6cc8905cf7891571d9fd5cfe644cd (HEAD -> main)
Author: itssyoma <itssyoma@gmail.com>
Date:   Wed Oct 11 09:58:47 2023 +0300

    add 2.txt and 3.txt
```

Рисунок 3 – Перезапись последнего коммита

```
itssyoma@Air-Sema megarepo_13 % git branch my_first_branch
itssyoma@Air-Sema megarepo_13 % git checkout my_first_branch
Switched to branch 'my_first_branch'
itssyoma@Air-Sema megarepo_13 % touch in_branch.txt
```

Рисунок 4 – Создание новой ветки и нового файла

```
itssyoma@Air-Sema megarepo_13 % git add .
itssyoma@Air-Sema megarepo_13 % git commit -m "add in_branch.txt"
[my_first_branch b684499] add in_branch.txt
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 .DS_Store
 create mode 100644 in_branch.txt
```

Рисунок 5 – Новый коммит

```
itssyoma@Air-Sema megarepo_13 % git checkout -b new_branch
Switched to a new branch 'new_branch'
```

Рисунок 5 – Переход на новую ветку

```
itssyoma@Air-Sema megarepo_13 % 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)
itssyoma@Air-Sema megarepo_13 % git merge my_first_branch
Updating 7ff09cf..b684499
Fast-forward
 .DS_Store     | Bin 0 -> 6148 bytes
 in_branch.txt |   0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 .DS_Store
 create mode 100644 in_branch.txt
itssyoma@Air-Sema megarepo_13 % git merge new_branch
Merge made by the 'ort' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)
itssyoma@Air-Sema megarepo_13 % git branch -d my_first_branch
Deleted branch my_first_branch (was b684499).
itssyoma@Air-Sema megarepo_13 % git branch -d new_branch
Deleted branch new_branch (was 4f02931).
```

Рисунок 6 – Переход на основную ветку, слияние, удаление веток

```
itssyoma@Air-Sema megarepo_13 % git branch branch_1
itssyoma@Air-Sema megarepo_13 % git branch branch_2
itssyoma@Air-Sema megarepo_13 % git checkout branch_1
Switched to branch 'branch_1'
itssyoma@Air-Sema megarepo_13 % git add .
itssyoma@Air-Sema megarepo_13 % git commit -m "fix 1 and 3"
[branch_1 13f4b3b] fix 1 and 3
 2 files changed, 2 insertions(+), 1 deletion(-)
itssyoma@Air-Sema megarepo_13 % git checkout branch_2
Switched to branch 'branch_2'
itssyoma@Air-Sema megarepo_13 % git add .
itssyoma@Air-Sema megarepo_13 % git commit -m "fix 1 and 3"
[branch_2 e746da2] fix 1 and 3
 2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 7 – Изменение и коммит файлов на новых ветках

```
itssyoma@Air-Sema megarepo_13 % git checkout branch_1
Switched to branch 'branch_1'
itssyoma@Air-Sema megarepo_13 % 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 8 – Слияние с конфликтами

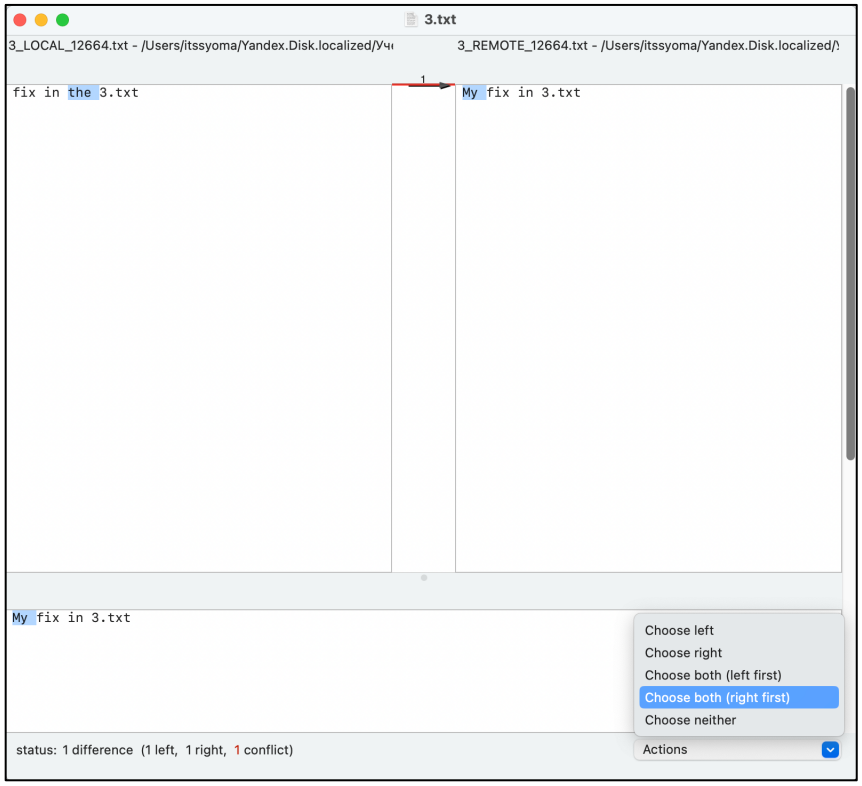

Рисунок 9 – исправление конфликта инструментом opendiff.

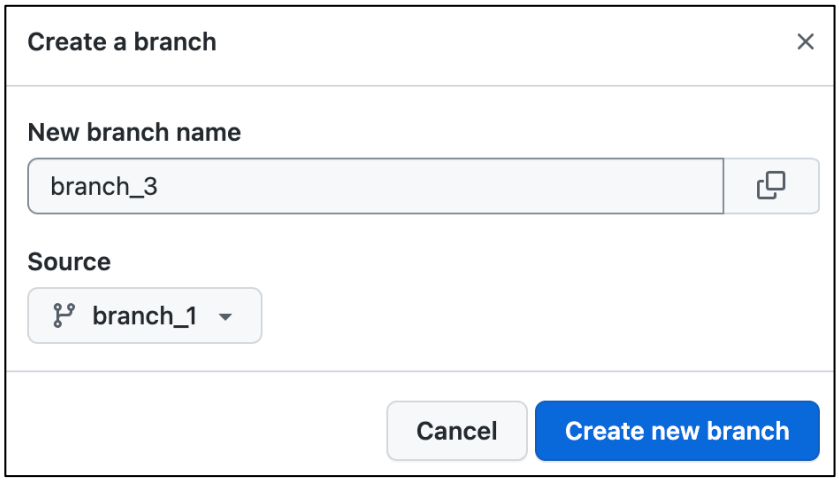

Рисунок 10 – создание удаленной ветки средствами GitHub.

```
itssyoma@Air-Sema megarepo_13 % git fetch --all
From https://github.com/itssyoma/megarepo_13
 * [new branch]      branch_3   -> origin/branch_3
itssyoma@Air-Sema megarepo_13 % git branch -vv
* branch_1 5ed4e5e 2 and 3 merge
  branch_2 e746da2 fix 1 and 3
  main     69a1d11 [origin/main: ahead 4] Merge branch 'new_branch'
itssyoma@Air-Sema megarepo_13 % git checkout --track origin/branch_3
branch 'branch_3' set up to track 'origin/branch_3'.
Switched to a new branch 'branch_3'
itssyoma@Air-Sema megarepo_13 % vi 2.txt
itssyoma@Air-Sema megarepo_13 % git add .
itssyoma@Air-Sema megarepo_13 % git commit -m "update 2.txt"
[branch_3 04003b9] update 2.txt
 1 file changed, 1 insertion(+)
itssyoma@Air-Sema megarepo_13 %
```

Рисунок 11 – создание ветки отслеживания, коммит измененного файла 2.txt.

```
itssyoma@Air-Sema megarepo_13 % git checkout branch_2
Switched to branch 'branch_2'
itssyoma@Air-Sema megarepo_13 % git rebase main
Current branch branch_2 is up to date.
```

Рисунок 12 – перемещение ветки main на ветку branch_2.

Ответы на контрольные вопросы

1. Что такое ветка?

Ветка в Git – это механизм, который позволяет создавать параллельные версии проекта. Каждая ветка содержит свою собственную историю коммитов, что позволяет работать над разными задачами одновременно без влияния на основную версию проекта.

2. Что такое HEAD?

HEAD – это указатель на текущий коммит в ветке. Он может указывать на любой коммит в истории проекта и обычно используется для перемещения по коммитам и создания новых веток.

3. Способы создания веток.

Существует несколько способов создания веток в Git:

– С помощью команды git branch <имя ветки> можно создать новую ветку на основе текущего HEAD.

– Команда git checkout -b <имя ветки> создаст новую ветку и автоматически переключит на нее.

– Используя команду git clone <URL репозитория>, можно склонировать удаленный репозиторий и автоматически создать локальную ветку.

– Используя инструменты GitHub.

4. Как узнать текущую ветку?

Чтобы узнать текущую ветку, можно использовать команду git branch или git status. В выводе будет указана активная ветка с символом * перед названием.

5. Как переключаться между ветками?

Для переключения между ветками можно использовать команду git checkout <имя ветки>.

6. Что такое удаленная ветка?

Удаленная ветка – это ветка, которая находится на удаленном репозитории. Она может быть создана локально и отправлена на удаленный репозиторий или создана на удаленном репозитории и затем склонирована локально.

7. Что такое ветка отслеживания?

Ветка отслеживания – это локальная ветка, которая связана с удаленной веткой. Это позволяет отслеживать изменения в удаленной ветке и отправлять свои изменения на нее.

8. Как создать ветку отслеживания?

Для создания ветки отслеживания нужно использовать команду git checkout -b <имя ветки> <имя удаленной ветки>.

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

Чтобы отправить изменения из локальной ветки в удаленную, нужно сначала добавить изменения в индекс с помощью команды git add, затем создать коммит с помощью git commit и отправить изменения на удаленный репозиторий с помощью команды git push <имя удаленного репозитория> <имя локальной ветки>.

10. В чем отличие команд git fetch и git pull?

Команда git fetch загружает изменения из удаленного репозитория, но не объединяет их с локальной веткой. Команда git pull загружает изменения и автоматически объединяет их с локальной веткой.

11. Как удалить локальную и удаленную ветки?

Чтобы удалить локальную ветку, можно использовать команду git branch -d <имя ветки>. Для удаления удаленной ветки нужно использовать команду git push <имя удаленного репозитория> --delete <имя ветки>.

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/10691 2/). 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 git-flow присутствуют два основных типа веток: master и develop. Кроме того, существуют вспомогательные ветки feature, release, hotfix и support.

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

1. Ветка master предназначена для хранения стабильной версии проекта. На нее могут быть сделаны только исправления ошибок (hotfix).

2. Ветка develop используется для разработки новых функций и исправления ошибок. Из нее создаются ветки feature для работы над отдельными задачами.

3. Ветки feature создаются для работы над конкретными задачами и после завершения объединяются с веткой develop.

4. Ветки release создаются для подготовки к выпуску новой версии проекта. В них могут быть сделаны последние исправления ошибок и обновления документации. После этого они объединяются с веткой develop и master.

5. Ветки hotfix создаются для срочного исправления критических ошибок в текущей версии проекта. Они объединяются с веткой master и develop.

6. Ветки support используются для поддержки старых версий проекта, которые все еще используются клиентами.

Недостатки git-flow:

1. Модель git-flow может быть сложной для проектов с небольшой командой разработчиков.

2. Для работы с моделью git-flow необходимо использовать специальные утилиты, что может быть неудобно для новых разработчиков.

3. Ветки в модели git-flow могут быстро накапливаться, что может привести к путанице и сложностям при объединении изменений.

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.

Программные средства с GUI для работы с Git предоставляют различные инструменты для управления ветками. Например, в программе SourceTree есть возможность создавать новые ветки, переключаться между ними, объединять и удалять ветки. Также есть возможность просмотреть список всех веток и их последние коммиты.