Лабораторная работа №3

По дисциплине: Основы программной инженерии

Духно Михаил

ПИЖ-б-о-20-1

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git commit -m "add 1.txt file"
[main 6432b69] add 1.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>
```

Рисунок 3.1 – Коммит добавления первого файла

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git commit -m "add 2.txt file"
[main 7153e26] add 2.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2.txt
```

Рисунок 3.2 – Коммит добавления второго файла

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git commit -m "add 3.txt file"
[main b0a77a0] add 3.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 3.txt
```

Рисунок 3.3 – Коммит добавления третьего файла

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git branch my_first_branch

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git branch
* main
  my_first_branch
```

Рисунок 3.4 – Создание новой ветки my_first_branch

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git checkout my_first_branch
Switched to branch 'my_first_branch'
```

Рисунок 3.5 – Переход на ветку my_first_branch

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git commit -m "Commit"
[my_first_branch a530526] Commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt
```

Рисунок 3.6 – Коммит добавления файла in_branch.txt

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>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)
```

Рисунок 3.7 – Переход на ветку master

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git branch new_branch

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git checkout new_branch
Switched to branch 'new_branch'

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>
```

Рисунок 3.8 – Создание и переход на ветку new_branch

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git commit -m "new row in the 1.txt file"
[new_branch f8da91a] new row in the 1.txt file
 1 file changed, 1 insertion(+)
```

Рисунок 3.9 – Коммит изменений файла 1.txt

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>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)

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git merge my_first_branch
Updating b0a77a0..a530526
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git merge new_branch
Merge made by the 'recursive' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>
```

Рисунок 3.10 – Переход на ветку master и слияние с остальными ветками

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git branch -d my_first_branch
Deleted branch my_first_branch (was a530526).

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git branch -d new_branch
Deleted branch new_branch (was f8da91a).

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>
```

Рисунок 3.11 – Удаление веток

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git branch branch_1

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git branch branch_2

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git branch
  branch_1
  branch_2
* main

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>
```

Рисунок 3.12 – Создание двух новых веток

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git commit -m "chanched 1.txt 3.txt files"
[branch_1 ec827c8] chanched 1.txt 3.txt files
 2 files changed, 3 insertions(+), 1 deletion(-)

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>
```

Рисунок 3.13 – Внесение изменений в файл и коммит этих изменений

```
C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git add .

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>git commit -m "Chanched 1.txt and 3.txt files"
[branch_2 1e4ae17] Chanched 1.txt and 3.txt files
 2 files changed, 2 insertions(+), 1 deletion(-)

C:\Users\я\Desktop\Папки\Программирование\Python\laboratory3>
```

Рисунок 3.14 – Переход на другую ветку и коммит изменений внесённых в файлы

Контрольные вопросы:

1. Что такое ветка?

Ветка в Git — это простой перемещаемый указатель на один из таких коммитов. По умолчанию, имя основной ветки в Git — master . Как только вы начнёте создавать коммиты, ветка master будет всегда указывать на последний коммит.

2. Что такое *HEAD*?

HEAD в Git - это указатель на текущую ссылку на ветвь, которая, в свою очередь, является указателем на последний сделанный вами коммит или последний коммит,

который был извлечен в ваш рабочий каталог. Это также означает, что это будет родитель следующего коммита, который вы делаете.

3. Способы создания веток.
`git checkout -b <имя ветки>`

4. Как узнать текущую ветку?
Узнать текущую ветку в Git можно путем просмотра списка веток через команду git branch

5. Как переключаться между ветками?
используйте команду `git checkout` для переключения между ветками

6. Что такое удаленная ветка?
Удалённые ветки действуют как закладки для напоминания о том, где ветки в удалённых репозиториях находились во время последнего подключения к ним. Они выглядят как [имя удал. репоз.]/[ветка]

7. Что такое ветка отслеживания?
Ветки слежения — это ссылки на определённое состояние удалённых веток.

8. Как создать ветку отслеживания?
git branch --set-upstream-to=<имя ветки>
9. Как отправить изменения из локальной ветки в удаленную ветку?
```
git push origin dev
git push --set-upstream origin dev
```

10. В чем отличие команд git fetch и git pull ?
`git pull` — это, по сути, команда `git fetch`, после которой сразу же следует `git merge`.
Команда `git fetch` получает изменения с сервера и сохраняет их в каталог `refs/remotes/`. Это действие (fetch) не влияет на локальные ветки и текущие изменения, просто изменения с удаленного сервера скачиваются в директорию локального репозитария.
Git merge сливает код в указаной репозитории

11. Как удалить локальную и удаленную ветки?
```
git branch -d для локальных
git push -d для удалённых
```

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/106912/). Какие основные типы веток присуствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?
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.