

Лабораторная работа №4

НКАбд-06-23

Улитина Мария Максимовна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Работа с репозиторием git	12
5	Выводы	14
	Список литературы	15

Список иллюстраций

4.1	giflow	8
4.2	giflow	8
4.3	node.js	8
4.4	node.js	8
4.5	commitizen	9
4.6	standard changelog	9
4.7	каталог	9
4.8	репозиторий	9
4.9	файл	9
4.10	git push	9
4.11	Node.js	10
4.12	отправка	10
4.13	git-flow	10
4.14	develop	10
4.15	хранилище	10
4.16	ветка	11
4.17	версия релиза	11
4.18	Журнал изменений	11
4.19	Основная ветка	11
4.20	данные на github	11
4.21	релиз на github	12
4.22	ветка для новой функциональности	12
4.23	объединение	12
4.24	релиз	12
4.25	журнал изменений	12
4.26	журнал изменений	12
4.27	основная ветка	13
4.28	данные на github	13
4.29	данные на github	13
4.30	данные на github	13
4.31	релиз на github	13

Список таблиц

1 Цель работы

Получение навыков правильной работы с репозиториями git.

2 Задание

1. Выполнить работу для тестового репозитория.
2. Преобразовать рабочий репозиторий в репозиторий с git-flow и conventional commits.

3 Теоретическое введение

Gitflow Workflow опубликована и популяризована Винсентом Дриссеном. Gitflow Workflow предполагает выстраивание строгой модели ветвления с учётом выпуска проекта. Данная модель отлично подходит для организации рабочего процесса на основе релизов. Работа по модели Gitflow включает создание отдельной ветки для исправлений ошибок в рабочей среде.

4 Выполнение лабораторной работы

Установим gitflow (рис. 4.1).

```
root@10:~# dnf copr enable elegos/gitflow
```

Рис. 4.1: gitflow

(рис. 4.2)

```
root@10:~# dnf install gitflow
```

Рис. 4.2: gitflow

Настроим node.js

(рис. 4.3)

```
root@10:~# pnpm setup
```

Рис. 4.3: node.js

(рис. 4.4)

```
root@10:~# source ~/.bashrc
```

Рис. 4.4: node.js

Установим программу, используемую для помощи в форматировании коммитов (рис. 4.5)


```
mmulitina@10:~/git-extended$ pnpm add -g commitizen
```

Рис. 4.5: commitizen

Установим программу, используемую для помощи в создании логов (рис. 4.6)

```
mmulitina@10:~/git-extended$ pnpm add -g standard-changelog
```

Рис. 4.6: standard changelog

Создадим каталог для репозитория и перейдем в него (рис. 4.7)

```
mmulitina@10:~$ mkdir git-extended  
mmulitina@10:~$ cd git-extended
```

Рис. 4.7: каталог

Создадим репозиторий (рис. 4.8)

```
mmulitina@10:~/git-extended$ git init
```

Рис. 4.8: репозиторий

Отправим файл в репозиторий (рис. 4.9)

```
mmulitina@10:~/git-extended$ git add README.md  
mmulitina@10:~/git-extended$ git commit -m "first commit"
```

Рис. 4.9: файл

Отправим изменения на сервер (рис. 4.10)

```
mmulitina@10:~/git-extended$ git branch -M main  
mmulitina@10:~/git-extended$ git remote add origin https://github.com/mmilitina/  
git-extended.git  
mmulitina@10:~/git-extended$ git push -u origin main
```

Рис. 4.10: git push

Создадим конфигурацию для пакетов Node.js (рис. 4.11)

```
mmulitina@10:~/git-extended$ pnpm init
```

Рис. 4.11: Node.js

Изменим конфигурацию и отправим файлы на сервер (рис. 4.12)

```
mmulitina@10:~/git-extended$ git add .
mmulitina@10:~/git-extended$ git cz
cz-cli@4.3.0, cz-conventional-changelog@3.3.0

? Select the type of change that you're committing: docs: Documentation only
  changes
? What is the scope of this change (e.g. component or file name): (press enter
to skip)
? Write a short, imperative tense description of the change (max 94 chars):
  (3) new
? Provide a longer description of the change: (press enter to skip)

? Are there any breaking changes? No
? Does this change affect any open issues? No
[main 949d964] docs: new
Committer: mmulitina <mmulitina@10> 0.2.15s
```

Рис. 4.12: отправка

Инициализируем git-flow (рис. 4.13)

```
mmulitina@10:~/git-extended$ git flow init
```

Рис. 4.13: git-flow

Проверим, что мы на ветке develop (рис. 4.14)

```
mmulitina@10:~/git-extended$ git branch
* develop
  main
```

Рис. 4.14: develop

Загрузим весь репозиторий в хранилище (рис. 4.15)

```
mmulitina@10:~/git-extended$ git push --all
```

Рис. 4.15: хранилище

Установим внешнюю ветку как вышестоящую для этой ветки (рис. 4.16)

```
mmulitina@10:~/git-extended$ git branch --set-upstream-to=origin/develop develop
```

Рис. 4.16: ветка

Создадим релиз с версией 1.0.0 (рис. 4.17)

```
mmulitina@10:~/git-extended$ git flow release start 1.0.0
```

Рис. 4.17: версия релиза

Создадим журнал изменений и добавим журнал изменений в индекс (рис. 4.18)

```
mmulitina@10:~/git-extended$ standard-changelog --first-release
✓ created CHANGELOG.md
✓ output changes to CHANGELOG.md
mmulitina@10:~/git-extended$ git add CHANGELOG.md
mmulitina@10:~/git-extended$ git commit -am 'chore(site):add changelog'
```

Рис. 4.18: Журнал изменений

Зальём релизную ветку в основную ветку (рис. 4.19)

```
mmulitina@10:~/git-extended$ git flow release finish 1.0.0
```

Рис. 4.19: Основная ветка

Отправим данные на github (рис. 4.20)

```
mmulitina@10:~/git-extended$ git push --all
Перечисление объектов: 6, готово.
Подсчет объектов: 100% (6/6), готово.
При сжатии изменений используется до 7 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (5/5), 562 байта | 562.00 КиБ/с, готово.
Всего 5 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com:mmulitina/git-extended.git
  949d964..7870585 develop -> develop
  949d964..0de43bf main -> main
mmulitina@10:~/git-extended$ git push --tags
```

Рис. 4.20: данные на github

Создадим релиз на github. Для этого будем использовать утилиты работы с github (рис. 4.21)

```
mmulitina@10:~/git-extended$ gh release create v1.0.0 -F CHANGELOG.md
```

Рис. 4.21: релиз на github

4.1 Работа с репозиторием git

Создадим ветку для новой функциональности (рис. 4.22)

```
mmulitina@10:~/git-extended$ git flow feature start feature_branch
```

Рис. 4.22: ветка для новой функциональности

По окончании разработки новой функциональности следующим шагом следует объединить ветку `feature_branch` с `develop` (рис. 4.23)

```
mmulitina@10:~/git-extended$ git flow featute finish feature_branch
usage: git flow <subcommand>
```

Рис. 4.23: объединение

Создадим релиз с версией 1.2.3: (рис. 4.24)

```
mmulitina@10:~/git-extended$ git flow release start 1.2.3
```

Рис. 4.24: релиз

Создадим журнал изменений (рис. 4.25)

```
mmulitina@10:~/git-extended$ standard-changelog
✓ output changes to CHANGELOG.md
```

Рис. 4.25: журнал изменений

Добавим журнал изменений в индекс (рис. 4.26)

```
mmulitina@10:~/git-extended$ git add CHANGELOG.md
mmulitina@10:~/git-extended$ git commit -am 'chore(site): update changelog'
```

Рис. 4.26: журнал изменений

Зальём релизную ветку в основную ветку (рис. 4.27)

```
mmulitina@10:~/git-extended$ git flow release finish 1.2.3
```

Рис. 4.27: основная ветка

Отправим данные на github (рис. 4.28)

```
mmulitina@10:~/git-extended$ git flow release finish 1.2.3
```

Рис. 4.28: данные на github

(рис. 4.29)

```
mmulitina@10:~/git-extended$ git push --all
```

Рис. 4.29: данные на github

(рис. 4.30)

```
mmulitina@10:~/git-extended$ git push --tags  
Перенесение объектов: 1 категор
```

Рис. 4.30: данные на github

Создадим релиз на github с комментарием из журнала изменений (рис. 4.31)

```
mmulitina@10:~/git-extended$ gh release create v1.2.3 -F CHANGELOG.md
```

Рис. 4.31: релиз на github

5 Выводы

В процессе выполнения лабораторной работы я получила навыки правильной работы с репозиториями git.

Список литературы

1. Лабораторная работа №4.