Отчёта по лабораторной работе

Лабораторная работа №4

Диана Садова Алексеевна

Содержание

# 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 Последовательность выполнения работы

## 3.1 Установка программного обеспечения

### 3.1.1 Установка git-flow

Linux

Fedora  
  
 Установка из коллекции репозиториев Copr (https://copr.fedorainfracloud.org/coprs/elegos/gitflow/):(рис. [-@fig:001]),(рис. [-@fig:002]).

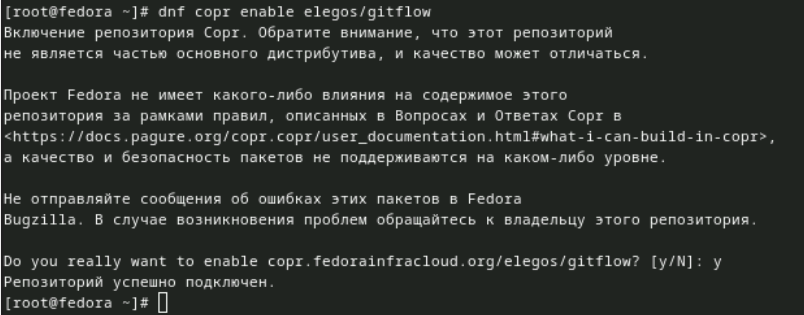


Рис. 1: Устонавливаем коллекции репозиториев Copr

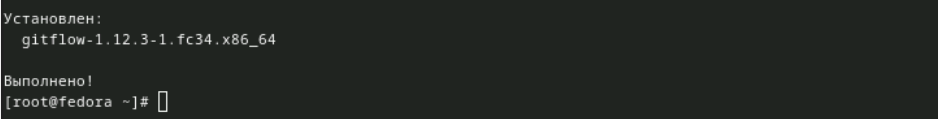


Рис. 2: Установка gitflow

После завершения этих програм можем двигатся дальше

### 3.1.2 Установка Node.js

На Node.js базируется программное обеспечение для семантического версионирования и общепринятых коммитов. (рис. 3),(рис. 4).

Fedora

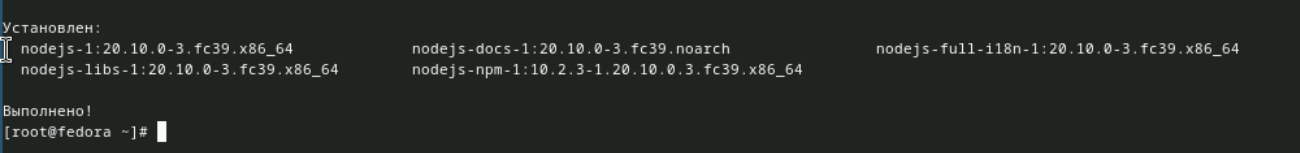


Рис. 3: Установка nodejs

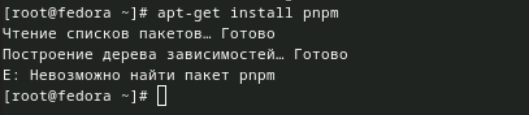


Рис. 4: Установка pnpm

Установка этих систем позволит выполнить данную лабораторную

### 3.1.3 Настройка Node.js

Для работы с Node.js добавим каталог с исполняемыми файлами, устанавливаемыми yarn, в переменную PATH.

Запустите:(рис. [-@fig:005]).

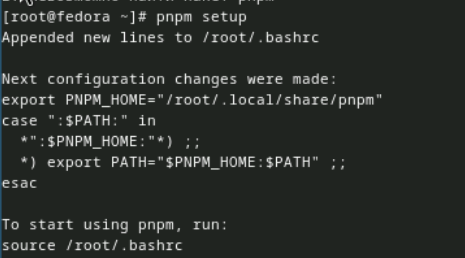


Рис. 5: Добавление каталога с исполняемыми файлами

Перелогиньтесь, или выполните:(рис. [-@fig:006]).

Выполняем программу source ~/.bashrc

Рис. 6: Выполняем программу source ~/.bashrc

### 3.1.4 Общепринятые коммиты

1)commitizen  
  
 Данная программа используется для помощи в форматировании коммитов.(рис. [-@fig:007]).

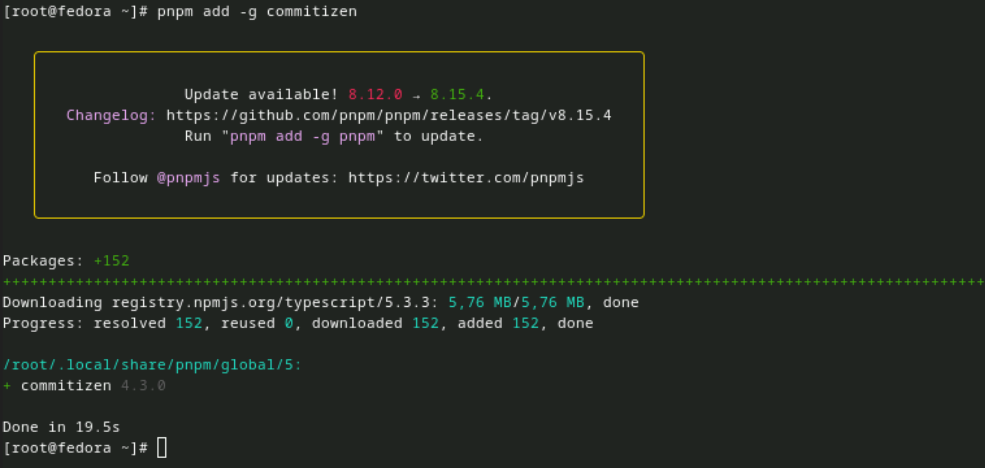


Рис. 7: Добавляем программу для форматирования коммитов

При этом устанавливается скрипт git-cz, который мы и будем использовать для коммитов.

После завершения установки переходим к пункту 2

2)standard-changelog  
  
 Данная программа используется для помощи в создании логов.(рис. [-@fig:008]).

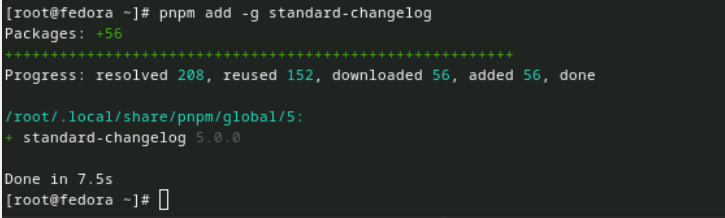


Рис. 8: Добавляем программу для помощи в создании логов

После завершения установки переходим к пункту 3

3)Практический сценарий использования git  
  
 1.Создание репозитория git  
  
 1.Подключение репозитория к github  
  
 Создайте репозиторий на GitHub. Для примера назовём его git-extended.рис. [-@fig:009]).

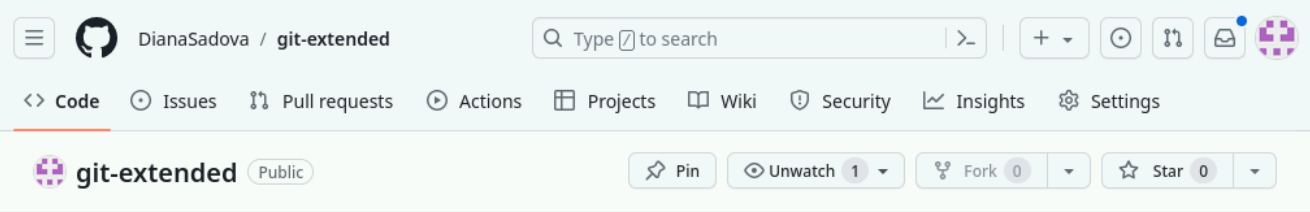


Рис. 9: Создаем новый репридиторий с именем git-extended

Делаем первый коммит и выкладываем на github:(рис. [-@fig:010]).

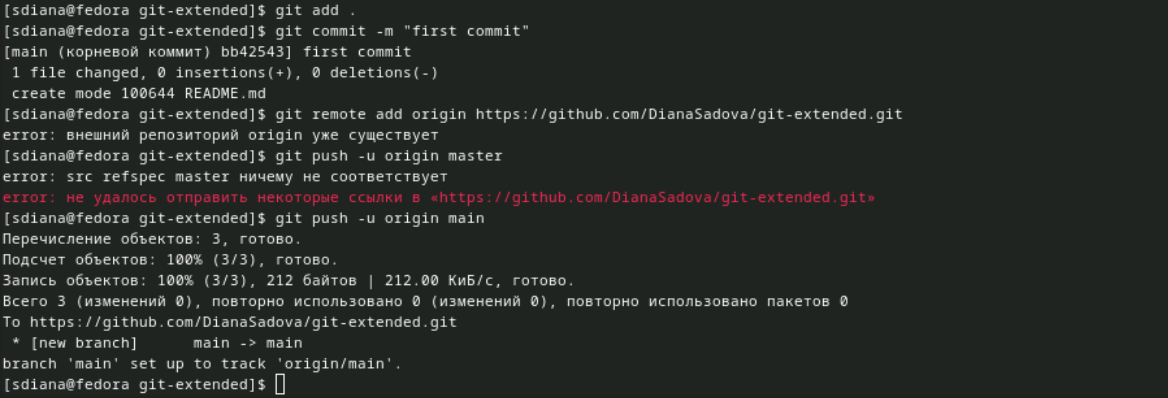


Рис. 10: Делаем коммит и отправляем на github

Для данного действия, я клонировала наш новый репридиторий и создала в нем файл README.md

2.Конфигурация общепринятых коммитов  
  
 Конфигурация для пакетов Node.js(рис. [-@fig:011]).

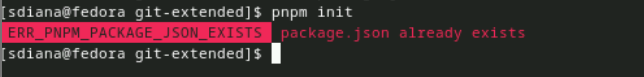


Рис. 11: Подключаем конфигурацию для пакетов Node.js

Необходимо заполнить несколько параметров пакета.  
 Название пакета.  
 Лицензия пакета. Список лицензий для npm: https://spdx.org/licenses/. Предлагается выбирать лицензию CC-BY-4.0.  
  
 Сконфигурим формат коммитов. Для этого добавим в файл package.json команду для формирования коммитов:  
  
 Таким образом, файл package.json приобретает вид:(рис. [-@fig:012]).

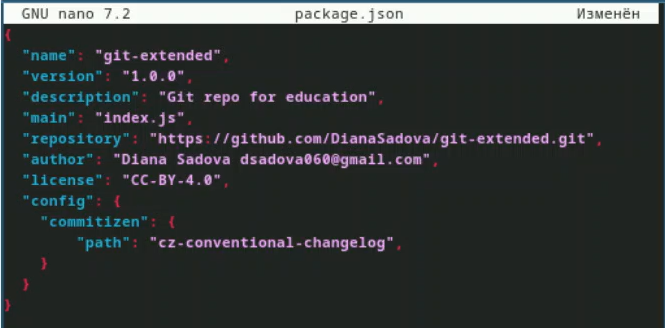


Рис. 12: Переписываем файл package.json

Данная процедура нам поможет при дальнейшей работе

Добавим новые файлы:(рис. [-@fig:013]).

Добавляем новые файлы

Рис. 13: Добавляем новые файлы

Выполним коммит:(рис. [-@fig:014]).

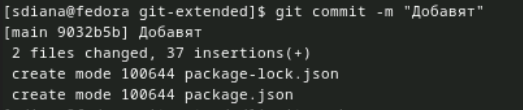


Рис. 14: Выполняем коммит

Выполняем коммит для нового файла

Отправим на github:(рис. [-@fig:015]).

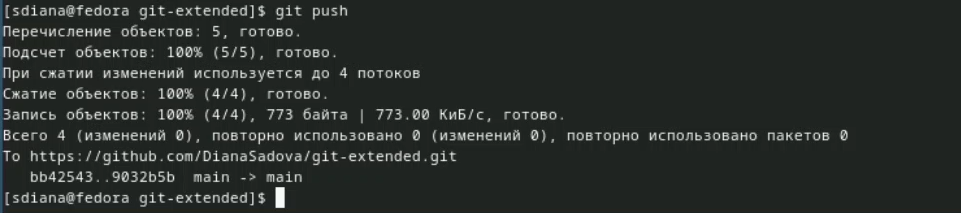


Рис. 15: Отправляем файл на github

3.Конфигурация git-flow  
  
 Инициализируем git-flow(рис. [-@fig:016]).

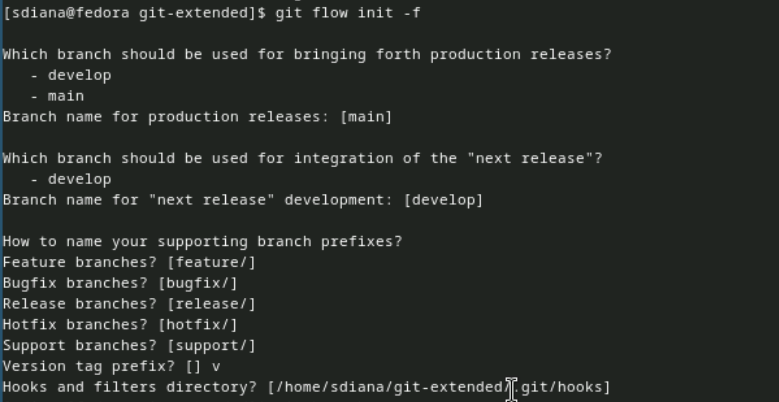


Рис. 16: Изменяем конфигурацию git-flow

Префикс для ярлыков установим в v.  
  
 Проверьте, что Вы на ветке develop:(рис. [-@fig:017]).

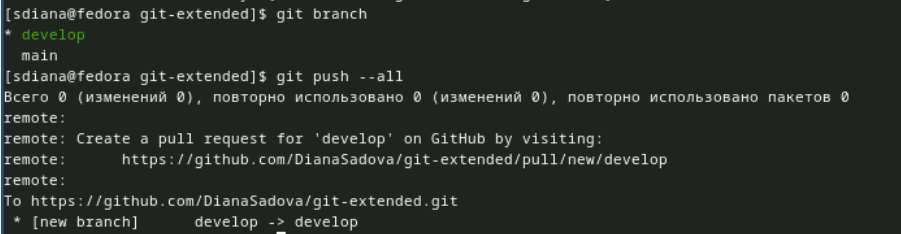


Рис. 17: Проверяем, что мы на ветке develop

Убедились, что мы на нужной ветке, ведь в последствии она нам понадобится

Загрузите весь репозиторий в хранилище:(рис. [-@fig:018]).

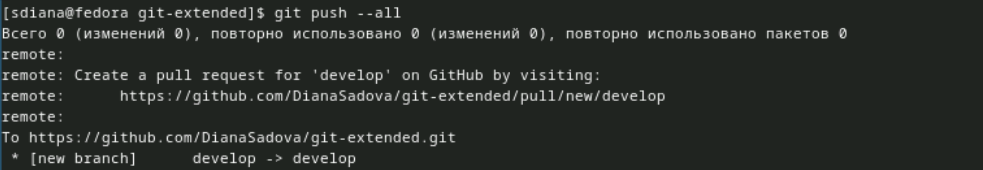


Рис. 18: Загружаем репозиторий в хранилище

Установите внешнюю ветку как вышестоящую для этой ветки:(рис. [-@fig:019]).

Установливаем внешнюю ветку

Рис. 19: Установливаем внешнюю ветку

Создадим релиз с версией 1.0.0(рис. [-@fig:020]).

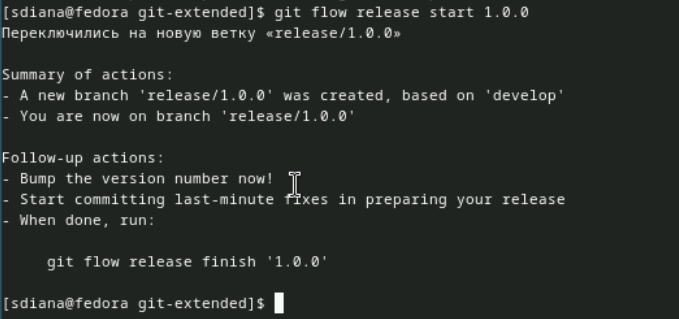


Рис. 20: Создаем релиз версии 1.0.0

Создадим журнал изменений(рис. [-@fig:021]).

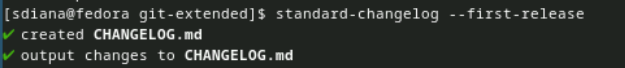


Рис. 21: Создаем журнал изменений

Что бы воспользоваься программой standard-changelog нужно было ее сначало установить (проверенно на собственном опыте)

Добавим журнал изменений в индекс(рис. [-@fig:022]).

Добавляем в индекс журнал изменений

Рис. 22: Добавляем в индекс журнал изменений

Зальём релизную ветку в основную ветку(рис. [-@fig:023]).

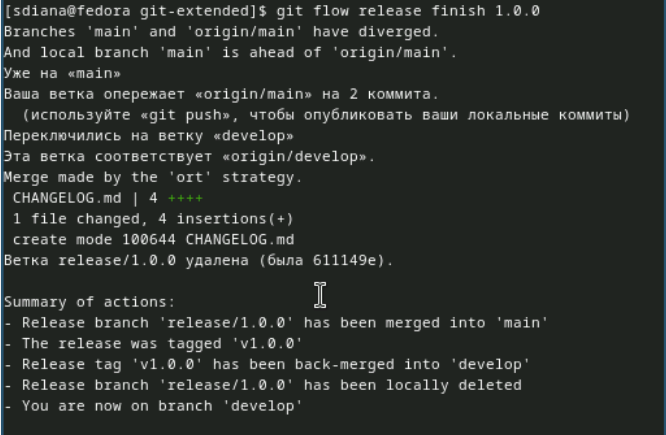


Рис. 23: Загружаем релизную ветку в основную ветку

Отправим данные на github(рис. [-@fig:024]),(рис. [-@fig:025]).

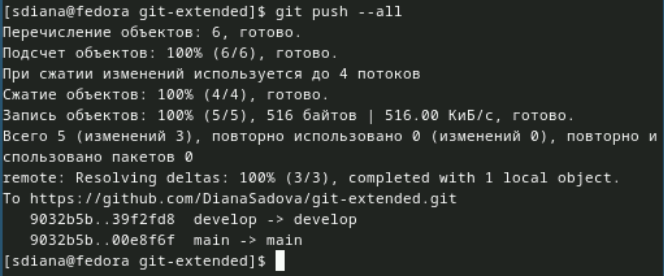


Рис. 24: Отправляем данные на github

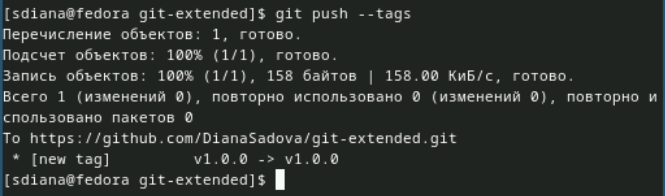


Рис. 25: Отправляем данные на github

Создадим релиз на github. Для этого будем использовать утилиты работы с github:(рис. [-@fig:026]).

Создаем релиз на github

Рис. 26: Создаем релиз на github

2.Работа с репозиторием git  
  
 1.Разработка новой функциональности  
  
 Создадим ветку для новой функциональности:(рис. [-@fig:027]).

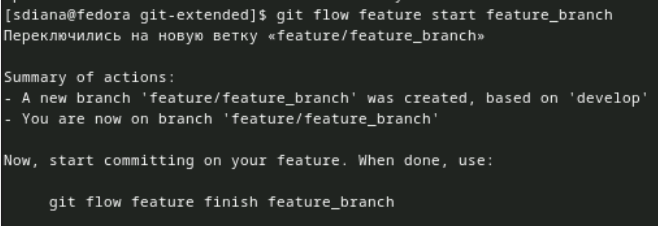


Рис. 27: Создадим новую ветку

Далее, продолжаем работу c git как обычно.  
  
 По окончании разработки новой функциональности следующим шагом следует объединить ветку feature\_branch c develop:(рис. [-@fig:028]).

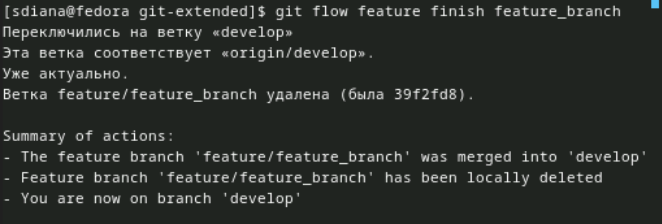


Рис. 28: ОбЪединяем две ветки

2.Создание релиза git-flow  
  
 Создадим релиз с версией 1.2.3:(рис. [-@fig:030]).

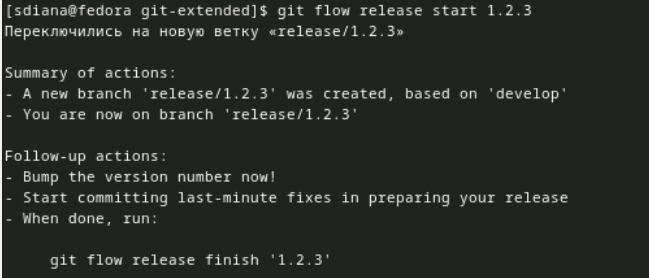


Рис. 29: Создаем новый релиз с версией 1.2.3

Обновите номер версии в файле package.json. Установите её в 1.2.3.,(рис. [-@fig:029]).

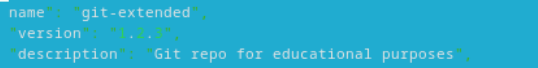


Рис. 30: Обновляем номер версии в файле package.json

Нам понадобится это обновление, так как без него не будут работать следующие программы

Создадим журнал изменений(рис. [-@fig:031]).

Создаем новый журнал изменений

Рис. 31: Создаем новый журнал изменений

Добавим журнал изменений в индекс(рис. [-@fig:032]),(рис. [-@fig:033]).

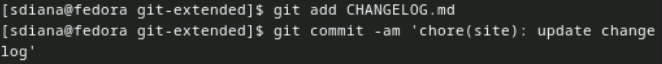


Рис. 32: Добавляем в индекс журнал изменений

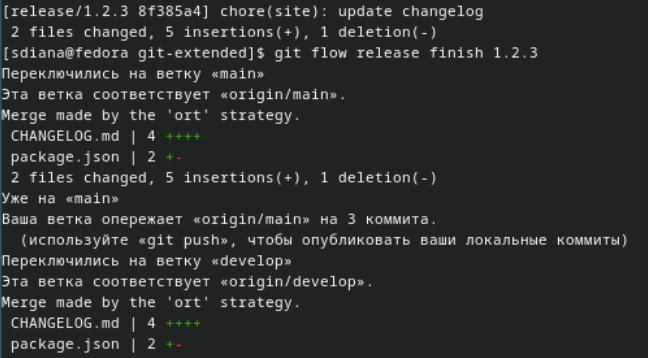


Рис. 33: Добавляем в индекс журнал изменений

Зальём релизную ветку в основную ветку(рис. [-@fig:034]).

Загружаем релизную ветку в основную ветку

Рис. 34: Загружаем релизную ветку в основную ветку

Отправим данные на github(рис. [-@fig:035]),(рис. [-@fig:036]).

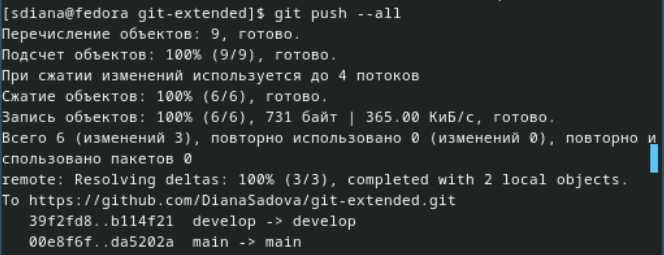


Рис. 35: Отправляем данные на github

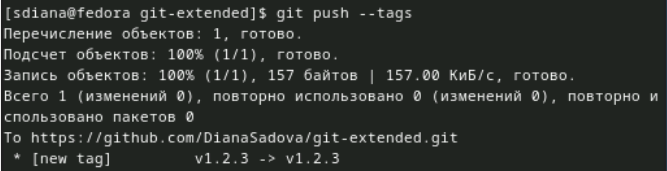


Рис. 36: Отправляем данные на github

Создадим релиз на github с комментарием из журнала изменений:(рис. [-@fig:037]).

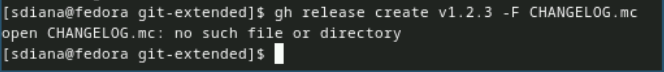


Рис. 37: Создаем новый релиз на github

# 4 Выводы

Получили навыки правильной работы с репозиториями git и освоили новые способы работы с ним

# Список литературы