Отчет по лабораторной работе №4

Операционные системы

Абронина Алиса Кирилловна

Содержание

Список иллюстраций

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 Выполнение лабораторной работы

# 4 Установка ПО. Установка git-glow, node.js. Настройка node.js.. Общепринятые коммиты.

Устанавливаем git-glow (рис. 1).

![Рис. 1: Установка git-glow](data:application/octet-stream;base64,)

Рис. 1: Установка git-glow

Устанавливаем node.js (рис. 2).

![Рис. 2: Установка node.js](data:application/octet-stream;base64,)

Рис. 2: Установка node.js

ДЛя работы с node.js добавим каталог с исполняемым файлами, устанавливаем yarn, в переменную path (рис. 3).

![Рис. 3: Настройка node.js](data:application/octet-stream;base64,)

Рис. 3: Настройка node.js

Настраиваем commitizen, standard-changelog (рис. 5).(рис. 4).

![Рис. 4: Настройка commitizen, standard-changelog](data:application/octet-stream;base64,)

Рис. 4: Настройка commitizen, standard-changelog

![Рис. 5: Настройка commitizen, standard-changelog](data:application/octet-stream;base64,)

Рис. 5: Настройка commitizen, standard-changelog

## 4.1 Практический сценарий использования git. Создание репозитория. Работа с репозиторием.

Создаем репозиторий на Github. Делаем первый коммит и выкладываем на github (рис. 6).

![Рис. 6: репозиторий](data:application/octet-stream;base64,)

Рис. 6: репозиторий

Конфигурация общепринятых коммитов. Для этого добавим в файл package.json команду для формирования коммитов (рис. 7).

![Рис. 7: Конфигурация коммитов](data:application/octet-stream;base64,)

Рис. 7: Конфигурация коммитов

Добавляем файлы, выполняем коммит, отправляем на github (рис. 8).

![Рис. 8: ОТправка изменений на github](data:application/octet-stream;base64,)

Рис. 8: ОТправка изменений на github

Инициализируем git-flow Префикс для ярлыков установим v.Проверяем, что мы работает на ветке develop. Загружаем весь репозиторий, устанавливаем внешнюю ветку как вышестоящюю для этой ветки. Создадим с версией 1.0.0 (рис. 9).(рис. 10). (рис. 11).

![Рис. 9: ч1](data:application/octet-stream;base64,)

Рис. 9: ч1

![Рис. 10: ч2](data:application/octet-stream;base64,)

Рис. 10: ч2

![Рис. 11: ч3](data:application/octet-stream;base64,)

Рис. 11: ч3

Создадим журнал изменений. ДОбавим журнал изменений в индекс. Зальем релизную ветку ветку в основную ветку. Отправим данные на github. Создадим релиз на github. Для этого будем использовать утилиты с github (рис. 12).(рис. 13).(рис. 14).

![Рис. 12: ч1](data:application/octet-stream;base64,)

Рис. 12: ч1

![Рис. 13: ч2](data:application/octet-stream;base64,)

Рис. 13: ч2

![Рис. 14: ч3](data:application/octet-stream;base64,)

Рис. 14: ч3

Создадим ветку для новой функциональности. Следующим шагом следует объединить ветки feature-branch с develop(рис. 15).

![Рис. 15: Создание и слияние веток](data:application/octet-stream;base64,)

Рис. 15: Создание и слияние веток

Создадим релиз с версией 1.2.3. Обновим номер версии в файле package.json и установим ее в 1.2.3 (рис. 16).

![Рис. 16: Изменение файла](data:application/octet-stream;base64,)

Рис. 16: Изменение файла

Создадим журнал изменений, добавим журнал изменений в индекс, зальем релизную ветку в основную ветку, отправляем данные на github, создадим релиз с комментарием из журнала изменений (рис. 17).(рис. 18). (рис. 19).

![Рис. 17: Ч1](data:application/octet-stream;base64,)

Рис. 17: Ч1

![Рис. 18: Ч2](data:application/octet-stream;base64,)

Рис. 18: Ч2

![Рис. 19: Ч3](data:application/octet-stream;base64,)

Рис. 19: Ч3

# 5 Выводы

Я получила навыки правильной работы с репозиториеми git