## 0.1 Front matter

title: “Отчет по лабораторной работе №4” subtitle: “Архитектура компьютера и операционные системы” author: “Ванюшкина Татьяна Валерьевна”

## 0.2 Generic otions

lang: ru-RU toc-title: “Содержание”

## 0.3 Bibliography

bibliography: bib/cite.bib csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

## 0.4 Pdf output format

toc: true # Table of contents toc-depth: 2 lof: true # List of figures lot: true # List of tables fontsize: 12pt linestretch: 1.5 papersize: a4 documentclass: scrreprt ## I18n polyglossia polyglossia-lang: name: russian options: - spelling=modern - babelshorthands=true polyglossia-otherlangs: name: english ## I18n babel babel-lang: russian babel-otherlangs: english ## Fonts mainfont: IBM Plex Serif romanfont: IBM Plex Serif sansfont: IBM Plex Sans monofont: IBM Plex Mono mathfont: STIX Two Math mainfontoptions: Ligatures=Common,Ligatures=TeX,Scale=0.94 romanfontoptions: Ligatures=Common,Ligatures=TeX,Scale=0.94 sansfontoptions: Ligatures=Common,Ligatures=TeX,Scale=MatchLowercase,Scale=0.94 monofontoptions: Scale=MatchLowercase,Scale=0.94,FakeStretch=0.9 mathfontoptions: ## Biblatex biblatex: true biblio-style: “gost-numeric” biblatexoptions: - parentracker=true - backend=biber - hyperref=auto - language=auto - autolang=other\* - citestyle=gost-numeric ## Pandoc-crossref LaTeX customization figureTitle: “Рис.” tableTitle: “Таблица” listingTitle: “Листинг” lofTitle: “Список иллюстраций” lotTitle: “Список таблиц” lolTitle: “Листинги” ## Misc options indent: true header-includes: -

# keep figures where there are in the text

* + # keep figures where there are in the text

# 1 Задание

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

# 2 Цель работы

Целью данной работы является получение навыков правильной работы с репозиториями git.

# 3 Техническое обеспечение

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. Последовательность действий при работе по модели Gitflow: Из ветки master создаётся ветка develop. Из ветки develop создаётся ветка release. Из ветки develop создаются ветки feature. Когда работа над веткой feature завершена, она сливается с веткой develop. Когда работа над веткой релиза release завершена, она сливается в ветки develop и master. Если в master обнаружена проблема, из master создаётся ветка hotfix. Когда работа над веткой исправления hotfix завершена, она сливается в ветки develop и master.

# 4 Выполнение лабораторной работы

1. Установка программного обеспечения

Переключаемся на роль супер-пользователя

(рис.1 **¿fig:002?**) ![Переключение на роль супер-пользователя](data:application/octet-stream;base64,) {#fig:002}

Устанавливаем из коллекции репозиториев Copr

(рис.2 **¿fig:003?**) ![Установка из коллекции репозиториев Copr](data:application/octet-stream;base64,) {#fig:003}

(рис.3 **¿fig:004?**) ![Установка из коллекции репозиториев Copr](data:application/octet-stream;base64,) {#fig:004}

Устанавливаем Node.js

(рис.4 **¿fig:005?**) ![Установка Node.js](data:application/octet-stream;base64,) {#fig:005}

(рис.5 **¿fig:006?**) ![Установка Node.js](data:application/octet-stream;base64,) {#fig:006}

Запускаем :

(рис.6 **¿fig:007?**) ![Запуск](data:application/octet-stream;base64,) {#fig:007}

Выполняем :

(рис.7 **¿fig:008?**) ![source ~/.bashrc](data:application/octet-stream;base64,) {#fig:008}

Общепринятые коммиты:

(рис.8 **¿fig:009?**) ![pnpm add -g commitizen](data:application/octet-stream;base64,) {#fig:009}

(рис.9 **¿fig:010?**) ![pnpm add -g standard-changelog](data:application/octet-stream;base64,) {#fig:010}

2.Практический сценарий использования git

Создаем репозиторий git-extended на GitHub

(рис.10 **¿fig:011?**) ![Создание репозитория](data:application/octet-stream;base64,) {#fig:011}

Клонируем его

(рис.11 **¿fig:012?**) ![Клонирование репозитория](data:application/octet-stream;base64,) {#fig:012}

Создание первого коммита

(рис.12 **¿fig:013?**) ![Создание первого коммита](data:application/octet-stream;base64,) {#fig:013}

(рис.13 **¿fig:015?**) ![Создание первого коммита](data:application/octet-stream;base64,) {#fig:015}

(рис.14 **¿fig:016?**) ![Создание первого коммита](data:application/octet-stream;base64,) {#fig:016}

Конфигурируем для пакетов Node.js

(рис.15 **¿fig:017?**) ![Конфигурация для пакетов Node.js](data:application/octet-stream;base64,) {#fig:017}

Сконфигурим формат коммитов. Для этого добавим в файл package.json команду для формирования коммитов:

(рис.16 **¿fig:018?**) ![Добавление комманды в файл](data:application/octet-stream;base64,) {#fig:018}

Добавляем новые файлы и выполняем коммит:

(рис.17 **¿fig:019?**) ![Добавление файлов и выполнение коммита](data:application/octet-stream;base64,) {#fig:019}

Отправляем файлы на гитхаб

(рис.18 **¿fig:020?**) ![Отправка файлов на гитхаб](data:application/octet-stream;base64,) {#fig:020}

Инициализируем git-flow

(рис.19 **¿fig:021?**) ![Инициализация git-flow](data:application/octet-stream;base64,) {#fig:021}

Проверяем, что мы на ветке develop:

(рис.20 **¿fig:022?**) ![Проверка ветки](data:application/octet-stream;base64,) {#fig:022}

Загружаем весь репозиторий в хранилище

(рис.21 **¿fig:023?**) ![Загрузка репозитория в хранилище](data:application/octet-stream;base64,) {#fig:023}

Устанавливаем внешнюю ветку как вышестоящую для этой ветки:

(рис.22 **¿fig:024?**) ![Внешняя ветка](data:application/octet-stream;base64,) {#fig:024}

Создадим релиз с версией 1.0.0

(рис.23 **¿fig:025?**) ![Создание релиза](data:application/octet-stream;base64,) {#fig:025}

Создадим журнал изменений

(рис.24 **¿fig:026?**) ![Создание журнала изменений](data:application/octet-stream;base64,) {#fig:026}

Добавим журнал изменений в индекс

(рис.25 **¿fig:027?**) ![Добавление журнала изменений в индекс](data:application/octet-stream;base64,) {#fig:027}

Зальём релизную ветку в основную ветку

(рис.26 **¿fig:028?**) ![Внешняя ветка](data:application/octet-stream;base64,) {#fig:028}

Отправим данные на github

(рис.27 **¿fig:029?**) ![Отправка данных](data:application/octet-stream;base64,) {#fig:029}

(рис.28 **¿fig:030?**) ![Отправка данных](data:application/octet-stream;base64,) {#fig:030}

Создадим релиз на github. Для этого будем использовать утилиты работы с github:

(рис.29 **¿fig:031?**) ![Создание релиза](data:application/octet-stream;base64,) {#fig:031}

1. Работа с репозиторием git

Создадим ветку для новой функциональности

(рис.30 **¿fig:032?**) ![Создание ветки для новой функциональности](data:application/octet-stream;base64,) {#fig:032}

Объединяем ветку feature\_branch c develop

(рис.31 **¿fig:033?**) ![Создание ветки для новой функциональности](data:application/octet-stream;base64,) {#fig:033}

Создадим релиз с версией 1.2.3

(рис.32 **¿fig:034?**) ![Создание ветки для новой функциональности](data:application/octet-stream;base64,) {#fig:034}

Создадим журнал изменений

(рис.33 **¿fig:035?**) ![Создание журнала изменений](data:application/octet-stream;base64,) {#fig:035}

Добавим журнал изменений в индекс

(рис.34 **¿fig:036?**) ![Добавление журнала изменений в индекс](data:application/octet-stream;base64,) {#fig:034}

Зальём релизную ветку в основную ветку и отправим данные на github

(рис.35 **¿fig:037?**) ![Отправка файлов](data:application/octet-stream;base64,) {#fig:037}

(рис.36 **¿fig:038?**) ![Отправка файлов](data:application/octet-stream;base64,) {#fig:038}

# 5 Выводы

Я получила навыков правильной работы с репозиториями git.

# Список литературы

Курс: Архитектура компьютеров и операционные системы. Раздел “Операционные системы” (02.03.00, УГСН) (rudn.ru)