

Научное программирование

Отче по лабораторной работе № 2: Markdown

Кейела Патачона- НПМмд-02-21

Содержание

| | | |
|----------|---------------------------------------|-----------|
| 1 | Цель работы | 5 |
| 2 | Теоретические сведения | 6 |
| 3 | Выполнение лабораторной работы | 7 |
| 4 | Выводы | 14 |
| 5 | Контрольные вопросы | 15 |

List of Tables

List of Figures

| | | |
|------|----------------------|----|
| 3.1 | рисунок 1 | 7 |
| 3.2 | рисунок 2 | 7 |
| 3.3 | рисунок 3 | 8 |
| 3.4 | рисунок 4 | 8 |
| 3.5 | рисунок 5 | 9 |
| 3.6 | рисунок 6 | 9 |
| 3.7 | рисунок 7 | 10 |
| 3.8 | рисунок 8 | 10 |
| 3.9 | рисунок 9 | 11 |
| 3.10 | рисунок 10 | 12 |
| 3.11 | рисунок 11 | 12 |
| 3.12 | рисунок 12 | 12 |

1 Цель работы

Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

2 Теоретические сведения

Вся теоритическая часть по использованию языка разметки Markdown была взята из инструкции по лабораторной работе №2 на сайте: <https://esystem.rudn.ru/pluginfile.php/1284/markdown.pdf>

3 Выполнение лабораторной работы

1. Создадим учётную запись на <https://github.com>.

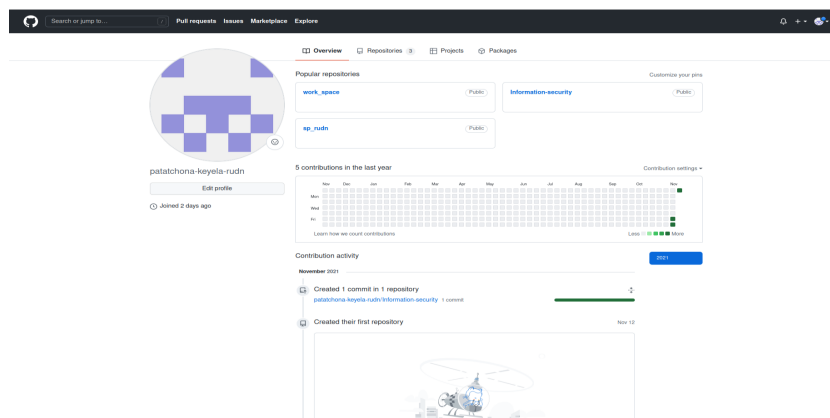


Figure 3.1: рисунок 1

2. Установим git на наш компьютер.



Figure 3.2: рисунок 2

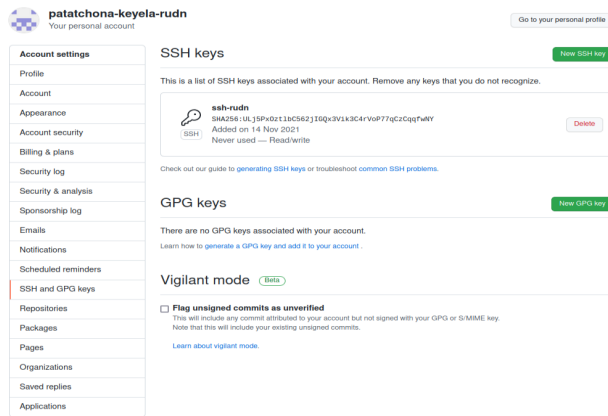


Figure 3.3: рисунок 3

3. Настроим систему контроля версий git, как это указано в инструкции к 1-ой лаборатной работе с использованием сервера репозиторииев <https://github.com/>.

Для этого необходимо сгенерировать пару ключей (приватный и открытый), а затем вставить их в SSH-ключи на github.

```
patatchona@ivan-VirtualBox:~/Laboratory$ cd -
patatchona@ivan-VirtualBox:~/Laboratory$ cd laboratory
patatchona@ivan-VirtualBox:~/Laboratory$ git init
патачона@ivan-VirtualBox:~/Laboratory$ git a /home/patatchona/laboratory/.git/
patatchona@ivan-VirtualBox:~/Laboratory$ echo "# Лабораторные работы" >> README.md
patatchona@ivan-VirtualBox:~/Laboratory$ git add README.md
patatchona@ivan-VirtualBox:~/Laboratory$ git commit -m "first commit"

*** Пожалуйста, скажите мне кто вы есть.

Запустите

git config --global user.email "you@example.com"
git config --global user.name "Ваше имя"

для указания идентификационных данных аккаунта по умолчанию.
Пропустите параметр --global для указания данных только для этого репозитория.

fatal: не удалось выполнить автоопределение адреса электронной почты (получено «patatchona@ivan-VirtualBox.(none)»)
patatchona@ivan-VirtualBox:~/Laboratory$ git config --global user.email "102216808@pfur.ru"
patatchona@ivan-VirtualBox:~/Laboratory$ git config --global user.name "Ваше имя"
patatchona@ivan-VirtualBox:~/Laboratory$ git config --global user.name "patatchona-keyela-rudn"
patatchona@ivan-VirtualBox:~/Laboratory$ git commit -m "first commit"
[master (корневой коммит) dcba722] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
patatchona@ivan-VirtualBox:~/Laboratory$ git remote add origin https://github.com/patatchona-keyela-rudn/sp_rudn.git
patatchona@ivan-VirtualBox:~/Laboratory$ git push -u origin master
Username for 'https://github.com': patatchona-keyela-rudn
Password for 'https://patatchona-keyela-rudn@github.com':
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more information.
fatal: Authentication failed for 'https://github.com/patatchona-keyela-rudn/sp_rudn.git/'
patatchona@ivan-VirtualBox:~/Laboratory$ git push -u origin master
Username for 'https://github.com': patatchona-keyela-rudn
Password for 'https://patatchona-keyela-rudn@github.com':
Пароль успешно введен: 3, готово.
Подсчет объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 253 байта | 253.00 Кбай/с, готово.
Всего 3 (изменения 0), повторно использовано 0 (изменения 0)
to https://github.com/patatchona-keyela-rudn/sp_rudn.git
 * [new branch] master -> master
Ветка «master» отслеживает внешние ветки «master» из «origin».
patatchona@ivan-VirtualBox:~/Laboratory$
```

Figure 3.4: рисунок 4

[illegible]

Результат проделанных операций представлен ниже.

[illegible]

master
1 branch
0 tags
Go to file
Add file
Code

patatchona-keyela-rudin task 1.5.3 88bafeb 3 minutes ago 2 commits

| | | |
|---------------|--------------|----------------|
| .gitignore | task 1.5.3 | 3 minutes ago |
| README.md | first commit | 10 minutes ago |
| legalcode.txt | task 1.5.3 | 3 minutes ago |

10

```

patatchona@ivan-VirtualBox:~/laboratory$ git flow init
Which branch should be used for bringing forth production releases?
- master
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
feature branches? [feature/]
bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [/home/patatchona/laboratory/.git/hooks]
patatchona@ivan-VirtualBox:~/laboratory$ git branch
* develop
  master
patatchona@ivan-VirtualBox:~/laboratory$ git flow release start 1.0.0
Переключено на новую ветку «release/1.0.0»

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish '1.0.0'

patatchona@ivan-VirtualBox:~/laboratory$ echo "1.0.0" >> VERSION
patatchona@ivan-VirtualBox:~/laboratory$ git add .
patatchona@ivan-VirtualBox:~/laboratory$ git commit -am 'chore(main): add version'
[release/1.0.0 cd0d471] chore(main): add version
1 file changed, 1 insertion(+)
create mode 100644 VERSION
patatchona@ivan-VirtualBox:~/laboratory$ git flow release finish 1.0.0
Переключено на ветку «master»
Ваша ветка обновлена в соответствии с «origin/master».
Merge made by the 'recursive' strategy.
  VERSION | 1
  1 file changed, 1 insertion(+)
  create mode 100644 VERSION
Уже на «master»
Ваша ветка опережает «origin/master» на 2 коммита.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)
Переключено на ветку «develop»
Merge made by the 'recursive' strategy.
  VERSION | 1
  1 file changed, 1 insertion(+)
  create mode 100644 VERSION
Ветка release/1.0.0 удалена (была cd0d471).

Summary of actions:
- Release branch 'release/1.0.0' has been merged into 'master'
- The release was tagged '1.0.0'
- Release tag '1.0.0' has been back-merged into 'develop'
- Release branch 'release/1.0.0' has been locally deleted
- You are now on branch 'develop'

```

Figure 3.9: рисунок 9

6. Первичная конфигурация

Добавим файл лицензии:

–`wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O`

Добавим шаблон игнорируемых файлов. Просмотрим список имеющихся шаблонов: –`curl -L -s https://www.gitignore.io/api/list`

Затем скачаем шаблон, например, для C:

–`curl -L -s https://www.gitignore.io/api/c » .gitignore`

Можно это же сделать через web-интерфейс на сайте <https://www.gitignore.io/>.

Добавим новые файлы:

–`git add .` Выполним коммит:

–`git commit -a` Отправим на github:

–`git push`

Результат проделанных операций представлен ниже.

```

patatchona@ivan-VirtualBox:~/laboratory$ git push --all
Username for 'https://github.com': patatchona-keyela-rudn
Password for 'https://patatchona-keyela-rudn@github.com':
Перечисление объектов: 6, готово.
Подсчет объектов: 100% (6/6), готово.
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (5/5), 494 байта | 494.00 КиБ/с, готово.
Всего 5 (изменения 3), повторно использовано 0 (изменения 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/patatchona-keyela-rudn/sp_rudn.git
  88bafcb..f42e271  master -> master
  * [new branch]      develop -> develop
patatchona@ivan-VirtualBox:~/laboratory$ git push --tags
Username for 'https://github.com': patatchona-keyela-rudn
Password for 'https://patatchona-keyela-rudn@github.com':
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 165 байтов | 165.00 КиБ/с, готово.
Всего 1 (изменения 0), повторно использовано 0 (изменения 0)
To https://github.com/patatchona-keyela-rudn/sp_rudn.git
  * [new tag]         1.0.0 -> 1.0.0
patatchona@ivan-VirtualBox:~/laboratory$

```

Figure 3.10: рисунок 10



Figure 3.11: рисунок 11

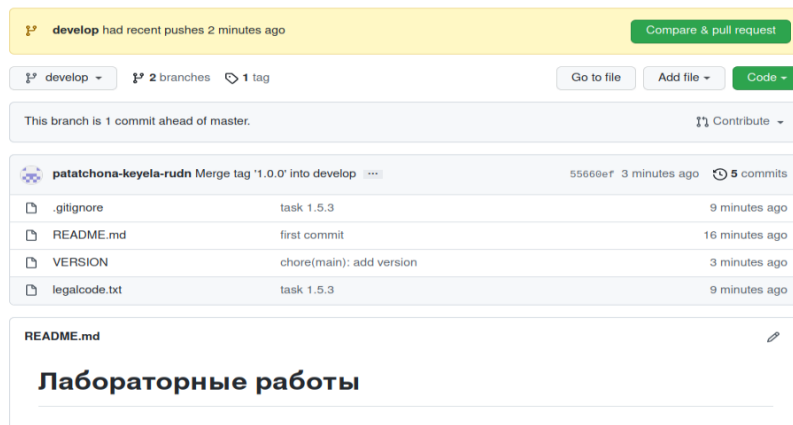


Figure 3.12: рисунок 12

7. Конфигурация git-flow

– Инициализируем git-flow

git flow init

Префикс для ярлыков установим в v.

– Проверьте, что Вы на ветке develop:

git branch

– Создадим релиз с версией 1.0.0

git flow release start 1.0.0

– Запишем версию:
echo "1.0.0" » VERSION – Добавим в индекс:
git add .
git commit -am 'chore(main): add version'
– Зальём релизную ветку в основную ветку
git flow release finish 1.0.0
– Отправим данные на github
git push -all
git push -tags
– Создадим релиз на github.

4 Выводы

Были изучена идеология и применение средств контроля версий. Также был освоен язык разметки Markdown.

5 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

- Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

- Хранилище (repository), или репозиторий, — место хранения всех версий и служебной информации. Commit («[трудовой] вклад», не переводится) — синоним версии; процесс создания новой версии. Рабочая копия (working copy) — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней).

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

- Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion. аспределенные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом

можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией. После внесения достаточного количества изменений в локальную копию они (изменения) отправляются на сервер. При этом сервер, чаще всего, выбирается условно, т.к. в большинстве DVCS нет такого понятия как “выделенный сервер с центральным репозиторием”.

4. Опишите действия с VCS при единоличной работе с хранилищем.

5. Опишите порядок работы с общим хранилищем VCS.

6. Каковы основные задачи, решаемые инструментальным средством git?

- Возврат к любой версии кода из прошлого. Просмотр истории изменений. Совместная работа без боязни потерять данные или затереть чужую работу.

7. Назовите и дайте краткую характеристику командам git.

- Наиболее часто используемые команды git:

- создание основного дерева репозитория:

`git init`

- получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull`

- отправка всех произведённых изменений локального дерева в центральный репозиторий:

`git push` – просмотр списка изменённых файлов в текущей директории: `git status`

- просмотр текущих изменений:

`git diff`

- добавить все изменённые и/или созданные файлы и/или каталоги: `git add .`

– добавить конкретные изменённые и/или созданные файлы и/или каталоги:
git add имена_файлов – удалить файл и/или каталог из индекса репозитория
(при этом файл и/или каталог остаётся в локальной директории):

git rm имена_файлов

– сохранение добавленных изменений:

– сохранить все добавленные изменения и все изменённые файлы: git commit
-am 'Описание коммита'

– сохранить добавленные изменения с внесением комментария через встроен-
ный редактор:

git commit

– создание новой ветки, базирующейся на текущей: git checkout -b имя_ветки

– переключение на некоторую ветку:

git checkout имя_ветки

– отправка изменений конкретной ветки в центральный репозиторий:

git push origin имя_ветки

– слияние ветки с текущим деревом:

git merge --no-ff имя_ветки

8. Приведите примеры использования при работе с локальным и удалённым
репозиториями.

- См пункты 1.3.3 -1.3.4

9. Что такое и зачем могут быть нужны ветви (branches)?

- Ветки нужны для того, чтобы программисты могли вести совместную работу
над проектом и не мешать друг другу при этом.

10. Как и зачем можно игнорировать некоторые файлы при commit?

- Игнорируемые файлы — это, как правило, артефакты сборки и файлы, ге-
нерируемые машиной из исходных файлов в вашем репозитории, либо файлы,
которые по какой-либо иной причине не должны попадать в коммиты. Для этого
нужно указать название все игнорируемых файлов в файле с названием .gitignore