

Цель работы


Целью данной работы является изучение идеологии и применение средств контроля версий.

Задание


- Сделайте отчёт по предыдущей лабораторной работе в формате Markdown.
- В качестве отчёта просьба предоставить отчёты в 3 форматах: pdf, docx и md (в архиве, поскольку он должен содержать скриншоты, Makefile и т.д.)

Выполнение лабораторной работы

1) В первую очередь, создадим учетную запись на <https://github.com>. (рис.1)

 Picture1




2) Настроим систему контроля версий git с использованием сервера репозитория: Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория(рис 2): `git config --global user.name "Имя Фамилия"`


`git config --global user.email "work@mail"`  Picture2 Для последующей идентификации

пользователя на сервере репозитория необходимо сгенерировать пару ключей, используя


команду: `ssh-keygen -C "Имя Фамилия <work@mail>"` (см. рис. 3)  Picture3 С помощью

команды `cat ~/.ssh/id_rsa.pub | xclip -sel clip` (Рис.4) скопируем из локальной консоли ключ в буфер обмена и вставляем ключ в появившееся на сайте поле. Для этого заходим в настройки, выбираем пункт «SSH and GPG keys»(рис.5), далее нажимаем «New SSH key» и


вставляем ключ в раздел «key»(Рис.6)  Picture4  Picture5  Picture6 В результате

создается ключ:  Picture7 Настройка системы контроля версий завершена.


3) Создаем репозиторий  Picture8 Задаем имя «laboratory2» и ставим галочку на пункте


«Add a README file». У нас создается репозиторий(рис.9)  Picture9 Теперь скачиваем


репозиторий на компьютер(Рис.10) с помощью команды: `git clone` и ссылки на репозиторий

 Picture10 Далее вводим команду `cd laboratory 2`, чтобы перейти в этот каталог.


Создаем каталог 2020-2021(`mkdir 2020-2021`) и переходим в него, далее создаем каталог «OS», переходим в него и создаем каталог «laboratory» и тоже переходим в него(Рис.11)

 Picture11 Делаем первый коммит и выкладываем его на гитхаб `git commit -m "first`


`commit"` (Рис.12)  Picture12 Создаем файл, используя команду `touch h.txt` и `git add .`

(Рис.13)  Picture13 Затем повторяем команду `git commit -m "first commit"` Используя

команду `git push`, отправим все произведенные изменения локального дерева в центральный

репозиторий. (Рис.14)  4) Первичная конфигурация. Добавим файл

лицензии(рис.15). Команда: wget


<https://creativecommons.org/licenses/by/4.0/legalcode.txt> -O LICENSE 

Добавляем шаблон игнорируемых файлов. Сначала посмотрим список имеющихся шаблонов:


curl -L -s <https://www.gitignore.io/api/list> (Рис.16)  Скачиваем шаблон. Я

взяла шаблон для C(Рис.17) Команда: curl -L -s <https://www.gitignore.io/api/c> >>


.gitignore Затем добавляем новые файлы, используя команду git add . Выполняем коммит


git commit -m "first commit" и отправляем на гитхаб git push.  5)

Конфигурация git-flow Для начала инициализируем git-flow Git flow init -f Префикс для



ярлыков устанавливаем в v.  Проверяем, что мы находимся на ветке develop,


командой git branch(Рис.19). Создаем релиз с версией 1.0.0 Git flow release start

1.0.0  Запишем версию(рис.20) echo "1.0.0" >> VERSION Добавим в индекс:

git add . git commit -am 'chore(main): add version'  Заливаем релизную

ветку в основную ветку, используя команду git flow release finish 1.0.0(Рис.21-22)

  Отправим данные на github git push --all git push --tags

(Рис.23)  Создаем релиз на гитхаб. Открываем «releases», в пункте «Target»

выбираем realese/1.0.0). (Рис.24)  У нас создается релиз(рис.25)



Выводы

Мы изучили идеологии и применение средств контроля версий.