Лабораторная работа №2

Научное программирование

Леонтьева Ксения Андреевна | НПМмд-02-23

Содержание

Список иллюстраций

1 Цель работы

Повторить процесс оформления отчётов с помощью легковесного языка разметки Markdown.

2 Теоретическое введение

Markdown - это язык разметки, используемый для форматирования документов всех типов, созданный Джоном Грубером и Аароном Шварцем в 2004 году, сегодня это один из самых популярных языков среди программистов. Многие идеи языка были позаимствованы из существующих соглашений по разметке текста в электронных письмах. Реализации языка Markdown преобразуют текст в формате Markdown в валидный, правильно построенный XHTML и заменяют левые угловые скобки («<») и амперсанды («&») на соответствующие коды сущностей. Первой реализацией Markdown стала написанная Грубером реализация на Perl, однако спустя некоторое время появилось множество реализаций от сторонних разработчиков. Реализация на Perl распространяется по лицензии типа BSD. Реализации Markdown на различных языках программирования включены (или доступны в качестве плагина) во многие системы управления содержимым.

Более подробно см. в [1]

3 Выполнение лабораторной работы

В данном разделе представлен отчет по предыдущей лабораторной работе.

Для начала была создана учетная запись и заполнены основные данные на сайте https://github.com (рис. fig. 3.1).

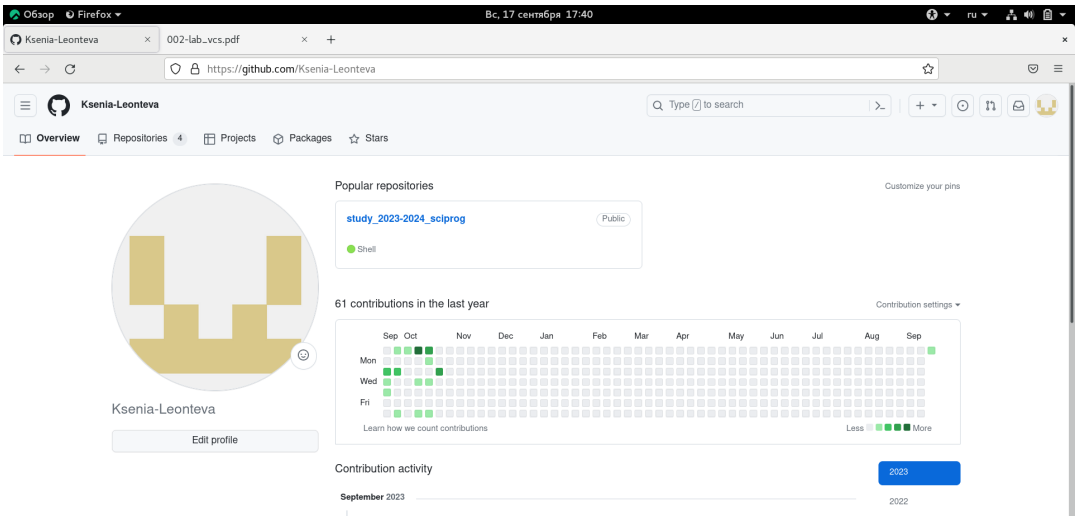

Рис. 3.1: Учетная запись на GitHub

Вручную установили программное обеспечение git-flow (рис. fig. 3.2).

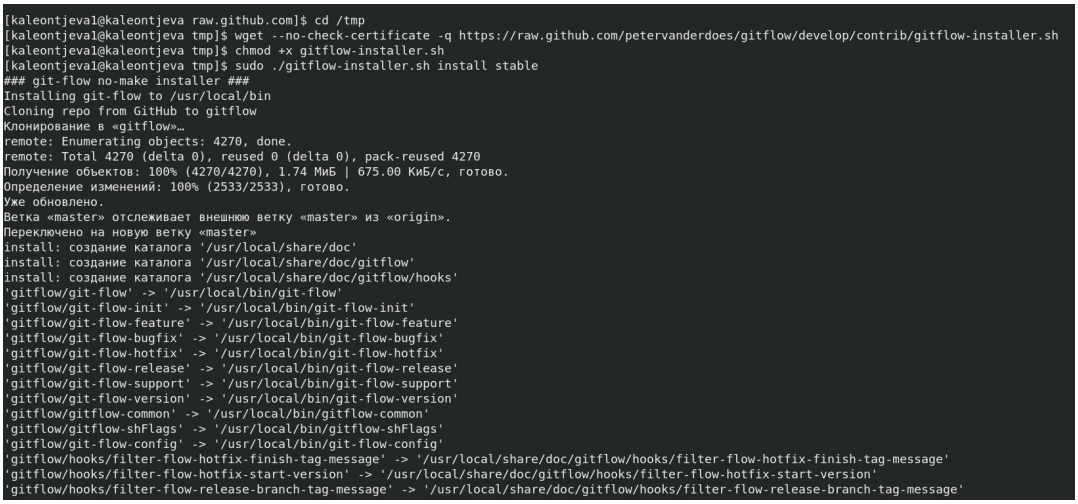

Рис. 3.2: Установка git-flow

Осуществили базовую настройку git (рис. fig. 3.3), а именно:

- задали имя и email владельца репозитория,
- настроили utf-8 в выводе сообщений git,
- настроили верификацию и подписание коммитов git,
- задали имя начальной ветки (master),
- параметр autocrlf,
- параметр safecrlf.

Также создали ключи ssh (рис. fig. 3.3 и рис. fig. 3.4):

- по алгоритму rsa с ключом размером 4096 бит,
- по алгоритму ed25519.

```
[kaleontjeva1@kaleontjeva ~]$ git config --global user.name "Kseniia Leonteva"
[kaleontjeva1@kaleontjeva ~]$ git config --global user.email "ksuleo23@gmail.com"
[kaleontjeva1@kaleontjeva ~]$ git config --global core.quotepath false
[kaleontjeva1@kaleontjeva ~]$ git config --global init.defaultBranch master
[kaleontjeva1@kaleontjeva ~]$ git config --global core.autocrlf input
[kaleontjeva1@kaleontjeva ~]$ git config --global core.safecrlf warn
[kaleontjeva1@kaleontjeva ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kaleontjeva1/.ssh/id_rsa): /home/kaleontjeva1/.ssh/id_rsa
/home/kaleontjeva1/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kaleontjeva1/.ssh/id_rsa.
Your public key has been saved in /home/kaleontjeva1/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:YiGPwpd7T7zdgbrIl7LpYh3BnvtOXcegjjWs9Jyc6JQ kaleontjeva1@kaleontjeva
The key's randomart image is:
+---[RSA 4096]----+
|                 |
|                 |
|    ...     .    |
| .   =o. . . o   |
|  o +.+oS = . o  |
|   o o++ % * .   |
|    ...oE.X .    |
|    oo+Oo+ . .   |
|   . o**O.. .    |
+----[SHA256]-----+
```

Рис. 3.3: Базовая настройка git и создание ключа ssh по алгоритму rsa

```
[kaleontjeva1@kaleontjeva ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/kaleontjeva1/.ssh/id_ed25519): /home/kaleontjeva1/.ssh/id_ed25519
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kaleontjeva1/.ssh/id_ed25519.
Your public key has been saved in /home/kaleontjeva1/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:mkuuLS6ZHHSClOI+TyfvlUQGSydKBCorYcA2yBN8alI kaleontjeva1@kaleontjeva
The key's randomart image is:
+--[ED25519 256]--+
|*o=o + .         |
|=XE.o =          |
|O+=. . o         |
|=*o . o          |
|*. o   .S        |
|.o.o ..o.        |
| .+++ +o         |
|  =..=..         |
|   o+++          |
+----[SHA256]-----+
[kaleontjeva1@kaleontjeva ~]$
```

Рис. 3.4: Создание ключа ssh по алгоритму ed25519

Сгенерировали ключ pgp и выбрали из предложенных опций необходимые (рис. fig. 3.5 и рис. fig. 3.6).

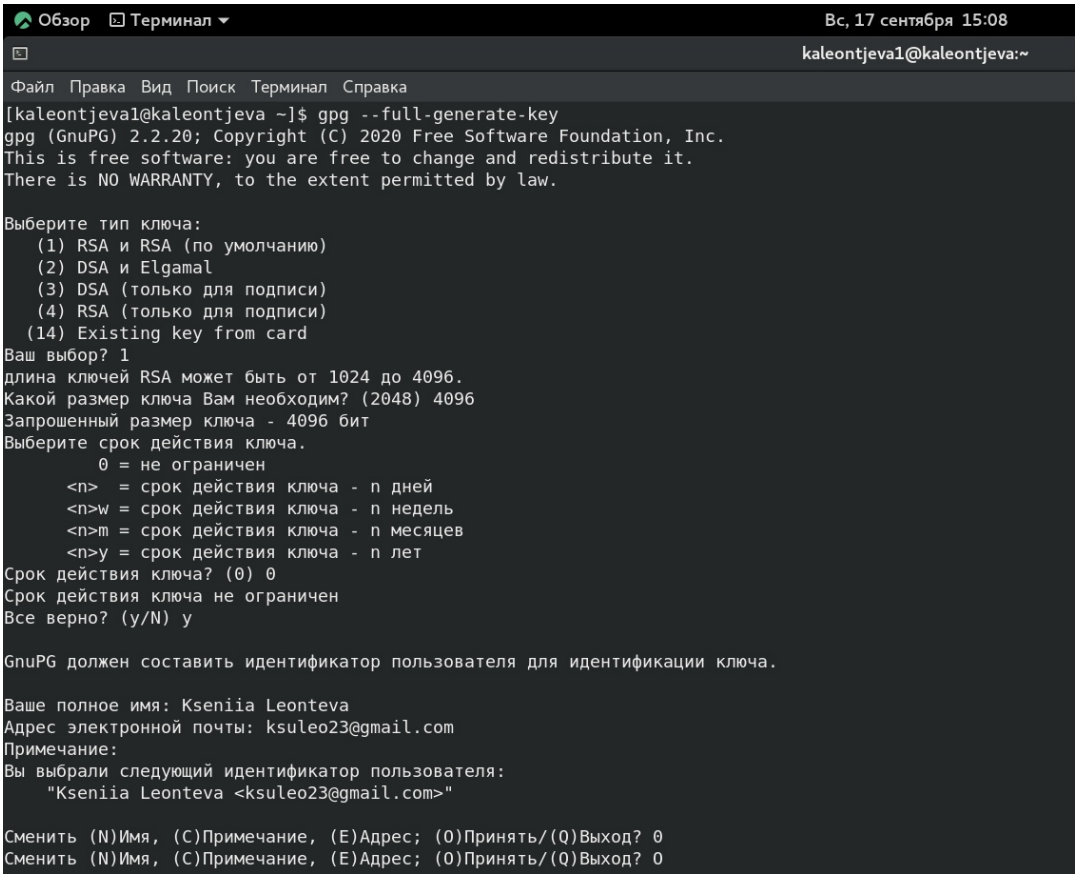

Рис. 3.5: Генерация ключа pgp

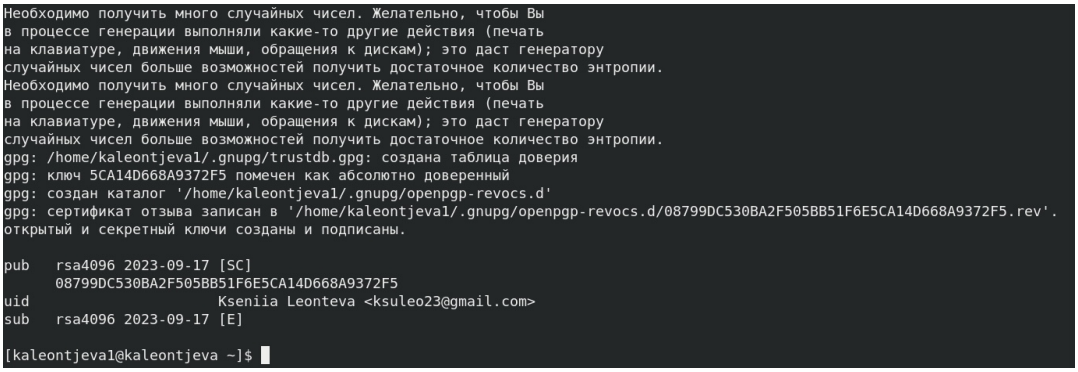

Рис. 3.6: Генерация ключа pgp

Выводим список ключей и копируем отпечаток приватного ключа (рис. fig. 3.7).

```
[kaleontjeva1@kaleontjeva ~]$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: глубина: 0  достоверных:   1  подписанных:   0  доверие: 0-, 0q, 0n, 0m, 0f, 1u
/home/kaleontjeva1/.gnupg/pubring.kbx
------------------------------------
sec   rsa4096/5CA14D668A9372F5 2023-09-17 [SC]
      08799DC530BA2F505BB51F6E5CA14D668A9372F5
uid                 [  абсолютно ] Kseniia Leonteva <ksuleo23@gmail.com>
ssb   rsa4096/377758E703B8BD1C 2023-09-17 [E]
```

Рис. 3.7: Вывод списка ключей

Копируем сгенерированный PGP ключ в буфер обмена (рис. fig. 3.8).

```
[kaleontjeva1@kaleontjeva ~]$ gpg --output public.pgp --armor --export ksuleo23@gmail.
com
[kaleontjeva1@kaleontjeva ~]$ cat prv.key
cat: prv.key: Нет такого файла или каталога
[kaleontjeva1@kaleontjeva ~]$ cat public.pgp
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGUG6/QBEACr+Omdim5+kaifwiITOmZ6Vac1z1qphE+tjCzmhHUXJjIxxIpI
RjGZwV3xp3pVxEXmFXcQUnQhARgif5TBMobJ82jMSeOwNbg6uu2NxjN2PNtJfyp9
yLg9CPGsGEVMO/kQ1K//aTQtSS3ScpfwZQ3ym8zPZHDJoUErj90GQjh+N7RHv5uG
VPjxcNUFbrwFkKl5J1CaqYSQxdvhX5NDN/OnNpCPhzrxhip+u7JwmkMJ7x5gBYnR
3YZs502S6egOsP7p75KPu4M5pcjDpQRHuc7u6MF2DPpfiDTjz1bKrUJaPde+Jdnk
Ya00S7pZbkm0qxWaBIkF68G7dxw5HcqRJLqz1I/PTMKRE1Ao2LCFI8i+f6DtqQxR
IuOjJPhDyMZISM7kO6J5Y3Fl5m4c+UUnk2WXc5x7KftnC28kvwOv50WhwMBoHwWj
mCIeQlxLy2mpGBy+raKvYuk0OfatKj96/QkwLKHp0UAbgZGJyHzPz847aTJeUE6J
EURBcqF0f9a2myQT40acZINfz5aG2lgU589PFth8jynp37pp4WNJuyTvzmd4X6ay
qiMl3URcBJoGa1Skuek49poLXR99M6xXbp3eJeEw6dOYnivE27S1bu63epM71QTd
cBs+5Cogr3flvQ8jzfdJrJEG+3SF1UIgOHYFbdOL4bw560laUB/ugbkpiwARAQAB
tCVLc2VuaWlhIExlb250ZXZhIDxrc3VsZW8yM0BnbWFpbC5jb20+iQJOBBMBCAA4
FiEECHmdxTC6L1BbtR9uXKFNZoqTcvUFAmUG6/QCGwMFCwkIBwIGFQoJCAsCBBYC
AwECHgECF4AACgkQXKFNZoqTcvXwMw//c/G1JFZ519vgIKoYYuNRd8eMN1ZH6xG4
oCZK33m7sQIdJCJBr8P4ZvzrKPL3R/KI2RmMsjwPhrMmaeJaJqpco9/hEa8it+6G
gkdqItHfahNtGjtCFyXxvmrXK8hh2fothUkfvUKSF/XQohOcxBqVpZyoqUcMlFG/
Wc29Jgia3Dhg2ipQoAuxPAepG9GKJUxV8IS1B49W6/uSlBVS+2wWVmzxbooITNRD
Qo9ccT3thWt+TBkmMQNi8GkEeUdtaOtWTdgcTStmqls/lcBOLxRfdFbuaXi0Z0xw
wsjIRL8Adq+nEC2stv4X+k7ACr9xM+52ZznKOstsKW40BUwg55lnE1QV2a3pRHNZ
C55I0qDJ0laDZfZBlIZ1xX/+3I8vC8asO6GvX92lDh0lUAxhm6tjUgnJXmwGebQY
CKXB5jaLn8ijmpKt3LDFx0TB1Nrc0KSCCrcSS0AYR/hx/FObc7ykjxgRU3NxyR+e
TwkwhC0pRhj9qG5gBIbxW20+iuOlGy1jJ7/RZMnTLJPiBqk2kqyYtGv9ns4+b53o
qpg8OX7GvR9Sfkqqz/QwwjglPhBzGj7pMtJg/zUBtP7XopgzGLGBjmNRFa6lUtN4
U/Jmi9KDFLDa9dhOGXJxhBzQ01hbSySN2SfzzbB8ZAWo9EgSZAI2u4MTkQ0PIbhB
```

Рис. 3.8: Копирование сгенерированного ключа PGP

Затем переходим в настройки GitHub, в раздел keys и, нажав на кнопку New GPG key, вставляем полученный ключ в поле ввода (рис. fig. 3.9).

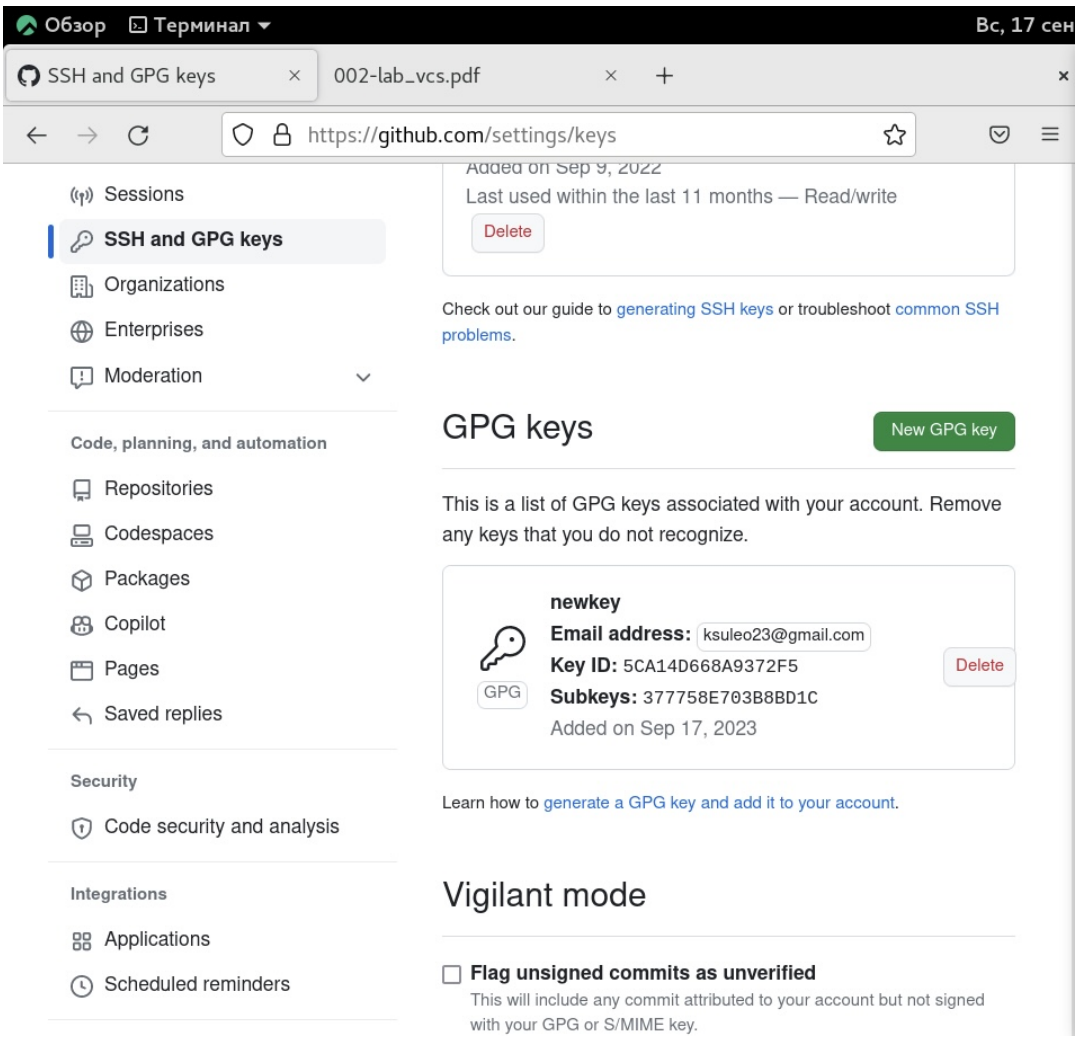

Рис. 3.9: Добавление gpg-ключа на GitHub

Настраиваем автоматические подписи коммитов git. Используя введенный email, указываем Git поменять его при подписи коммитов (рис. fig. 3.10).

```
[kaleontjeva1@kaleontjeva ~]$ git config --global user.signingkey ksuleo23@gmail.com
[kaleontjeva1@kaleontjeva ~]$ git config --global commit.gpgsign true
[kaleontjeva1@kaleontjeva ~]$ git config --global gpg.program $(which gpg2)
[kaleontjeva1@kaleontjeva ~]$
```

Рис. 3.10: Настройка автоматических подписей коммитов git

Устанавливаем и настраиваем gh. Отвечаем на наводящие вопросы утилиты и авторизуемся через браузер (рис. fig. 3.11 - рис. fig. 3.13).

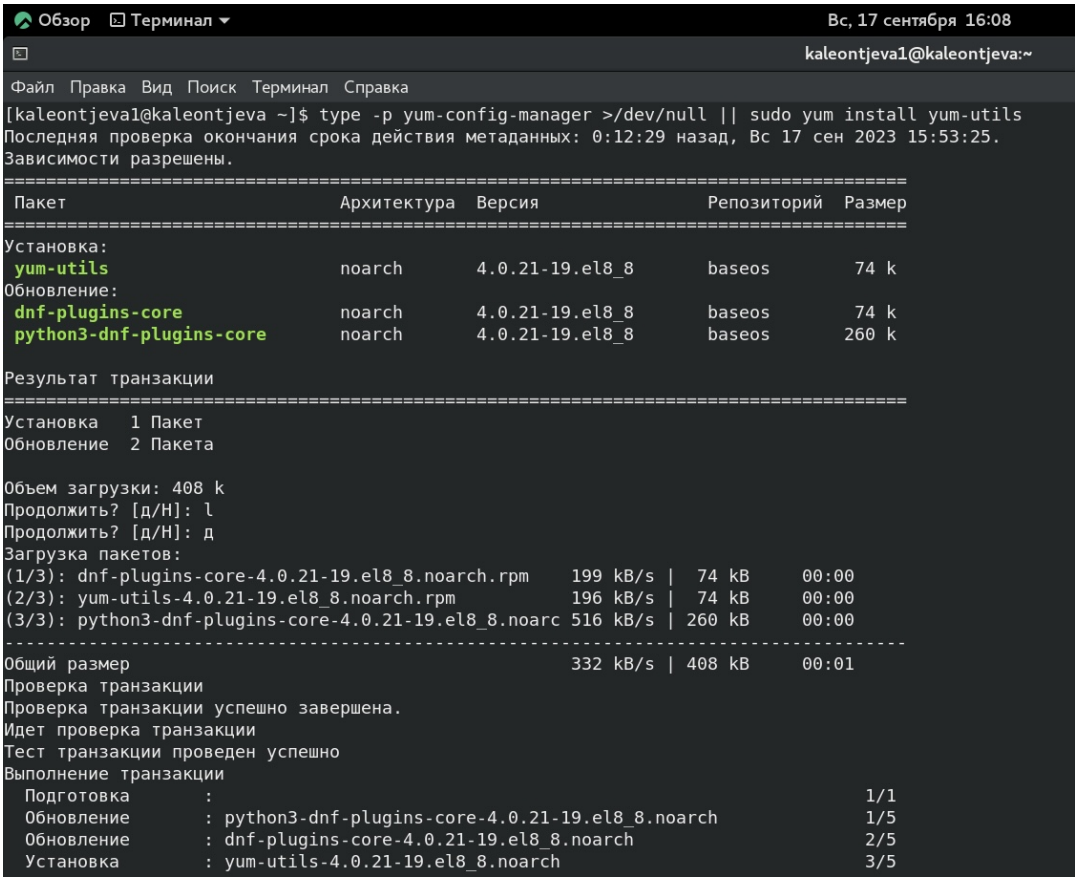

Рис. 3.11: Установка gh

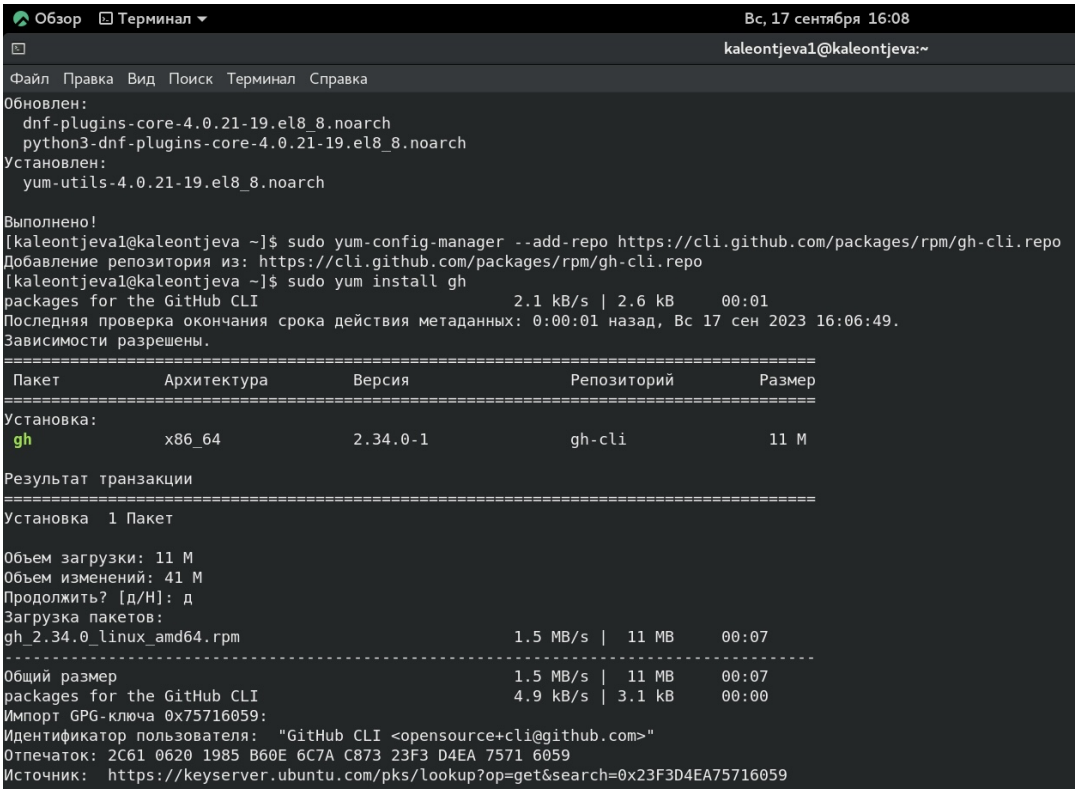

Рис. 3.12: Установка gh

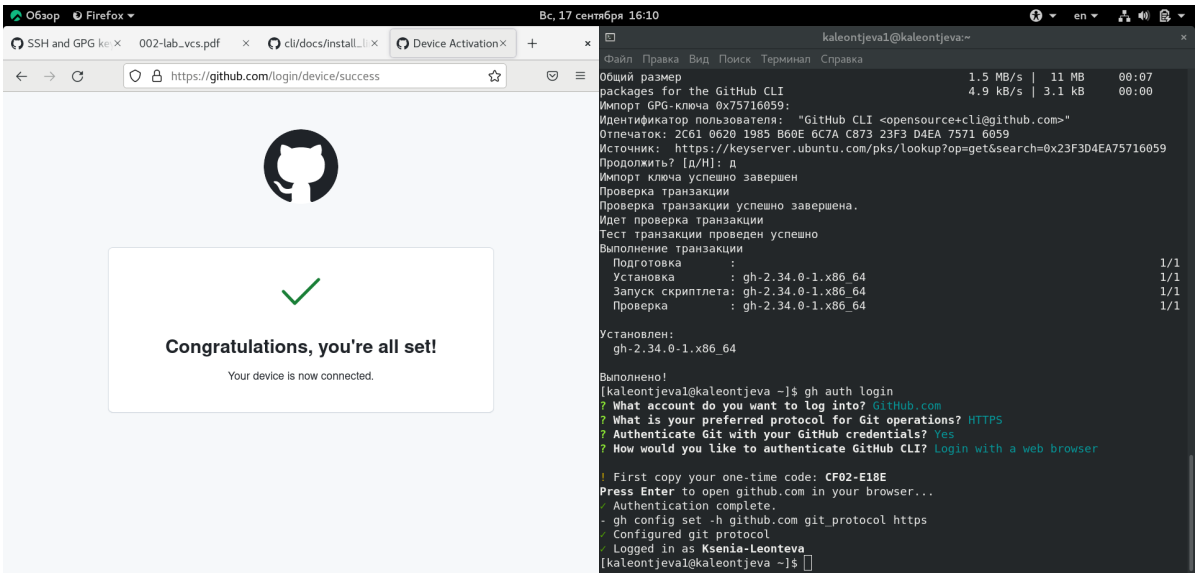

Рис. 3.13: Настройка gh и авторизация

Создаем репозиторий курса на основе шаблона (рис. fig. 3.14), параллельно сгенерировав и добавивив на GitHub еще один ssh-ключ (рис. fig. 3.15 и рис.

fig. 3.16).

```
[kaleontjeva1@kaleontjeva ~]$ cd ~/work/study
[kaleontjeva1@kaleontjeva study]$ mkdir -p 2023-2024/"Научное программирование"
[kaleontjeva1@kaleontjeva study]$ cd 2023-2024/"Научное программирование"
[kaleontjeva1@kaleontjeva Научное программирование]$ gh repo create study_2023-2024_sc
iprog-intro--template=yamadharma/course-directory-student-template --public
HTTP 404: Not Found (https://api.github.com/users/study_2023-2024_sciprog-intro--templ
ate=yamadharma)
[kaleontjeva1@kaleontjeva Научное программирование]$ gh repo create study_2023-2024_sc
iprog--template=yamadharma/course-directory-student-template --public
HTTP 404: Not Found (https://api.github.com/users/study_2023-2024_sciprog--template=ya
madharma)
[kaleontjeva1@kaleontjeva Научное программирование]$ gh repo create study_2023-2024_sc
iprog --template=yamadharma/course-directory-student-template --public
✓ Created repository Ksenia-Leonteva/study_2023-2024_sciprog on GitHub
```

Рис. 3.14: Создание репозитория курса

```
[kaleontjeva1@kaleontjeva Научное программирование]$ ssh-keygen -C "Kseniia Leonteva <
ksuleo23@gmail.com>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kaleontjeva1/.ssh/id_rsa):
/home/kaleontjeva1/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kaleontjeva1/.ssh/id_rsa.
Your public key has been saved in /home/kaleontjeva1/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:T9SJ94A/aWhP7ca8OVbfm1GGt9Kzk2c7GhuRUkV+sLE Kseniia Leonteva <ksuleo23@gmail.co
m>
The key's randomart image is:
+---[RSA 3072]----+
|              +o |
|           + .o= |
|          + =.E..|
|         . +.=...|
|        S +.*o+ +|
|         + +.=oo+|
|          . .+=*+|
|             .B*X|
|             ++B*|
+----[SHA256]-----+
[kaleontjeva1@kaleontjeva Научное программирование]$ cat ~/.ssh/id_rsa.pub | xclip -se
l clip
bash: xclip: команда не найдена...
Установить пакет «xclip», предоставляющий команду «xclip»? [N/y] y
```

Рис. 3.15: Генерация ssh-ключа

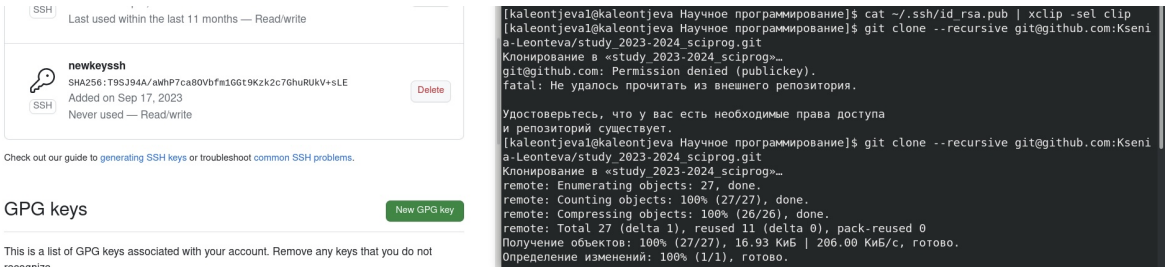

Рис. 3.16: Добавление ssh-ключа на GitHub и завершение создания репозитория курса

Настраиваем каталог курса (рис. fig. 3.17 и рис. fig. 3.18):

- переходим в каталог курса,
- удаляем лишние файлы,
- создаем необходимые каталоги,
- отправляем файлы на сервер.

```
[kaleontjeva1@kaleontjeva Научное программирование]$ cd sciprog
[kaleontjeva1@kaleontjeva sciprog]$ rm package.json
[kaleontjeva1@kaleontjeva sciprog]$ make COURSE=sciprog
[kaleontjeva1@kaleontjeva sciprog]$ git add .
[kaleontjeva1@kaleontjeva sciprog]$ git commit -am 'feat(main): make course structure'
[master 39f1839] feat(main): make course structure
 314 files changed, 86648 insertions(+), 14 deletions(-)
 create mode 100644 labs/README.md
 create mode 100644 labs/README.ru.md
 create mode 100644 labs/lab1/presentation/Makefile
 create mode 100644 labs/lab1/presentation/image/kulyabov.jpg
 create mode 100644 labs/lab1/presentation/presentation.md
 create mode 100644 labs/lab1/report/Makefile
 create mode 100644 labs/lab1/report/bib/cite.bib
 create mode 100644 labs/lab1/report/image/placeimg_800_600_tech.jpg
 create mode 100644 labs/lab1/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
```

Рис. 3.17: Настройка каталога курса

```
[kaleontjeva1@kaleontjeva sciprog]$ git push
Перечисление объектов: 41, готово.
Подсчет объектов: 100% (41/41), готово.
Сжатие объектов: 100% (31/31), готово.
Запись объектов: 100% (40/40), 343.21 КиБ | 2.84 МиБ/с, готово.
Всего 40 (изменений 5), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To github.com:Ksenia-Leonteva/study_2023-2024_sciprog.git
   aaadffb..39f1839  master -> master
[kaleontjeva1@kaleontjeva sciprog]$
```

Рис. 3.18: Настройка каталога курса

4 Контрольные вопросы

1. Система контроля версий (Version Control System, VCS) — это программное обеспечение, которое помогает отслеживать изменения в файловой системе и эффективно управлять версиями файлов и кода в проекте. Она позволяет разработчикам работать над проектами совместно, отслеживать, комментировать и объединять свои изменения.

2. Хранилище — место, куда помещается документ после внесения в него нужных правок. Оно является местом хранения служебной информации и всех версий документов. Commit — это пакет изменений, хранящий информацию с добавленными, отредактированными или удаленными файлами (в Git — это команда для записи индексированных изменений в репозиторий Git). История (в Git) — точный реестр всех коммитов, содержащих произведенные с файлами изменения. Рабочая копия — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.

3. 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.

Децентрализованные (распределенные) 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”. Две наиболее известные DVCS – это Git, Bazaar, Mercurial.

В отличие от централизованных, в распределенных системах контроля версий центральный репозиторий не является обязательным.

4. Создание локального репозитория. Делается предварительная конфигурация:

git config –global user.name “Имя Фамилия”
git config –global user.email “work@mail”
Настраивается utf-8 в выводе сообщений git:
git config –global quotepath false

Для инициализации локального репозитория, расположенного, например, в каталоге ~/tutorial, необходимо ввести в командной строке:

cd
mkdir tutorial
cd tutorial
git init

5. Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый):

ssh-keygen -C “Имя Фамилия work@mail”

Ключи сохраняться в каталоге ~/.ssh/. Скопировав из локальной консоли ключ в буфер обмена:

cat ~/.ssh/id_rsa.pub | xclip -sel clip
вставляем ключ в появившееся на сайте поле.

6. Основные задачи, решаемые инструментальным средством git: обеспечение удобной командной работы над проектом и хранение информации о всех изменениях в проекте.

7. Наиболее часто используемые команды git:

- git init – создание основного дерева репозитория,
- git pull – получение обновлений (изменений) текущего дерева из центрального репозитория,
- git push – отправка всех произведённых изменений локального дерева в центральный репозиторий,
- git status – просмотр списка изменённых файлов в текущей директории,
- git diff – просмотр текущих изменения,
- git add . – добавить все изменённые и/или созданные файлы и/или каталоги,
- git add имена_файлов – добавить конкретные изменённые и/или созданные файлы и/или каталоги,
- git rm имена_файлов – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории),
- git commit -am ‘Описание коммита’ – сохранить все добавленные изменения и все изменённые файлы,
- git commit – сохранить добавленные изменения с внесением комментария через встроенный редактор,
- git checkout -b имя_ветки – создание новой ветки, базирующейся на текущей,

- git checkout имя_ветки – переключение на некоторую ветку,
- git push origin имя_ветки – отправка изменений конкретной ветки в центральный репозиторий,
- git merge –no-ff имя_ветки – слияние ветки с текущим деревом,
- git branch -d имя_ветки – удаление локальной уже слитой с основным деревом ветки,
- git branch -D имя_ветки – принудительное удаление локальной ветки,
- git push origin :имя_ветки – удаление ветки с центрального репозитория.

8. Создание тестового файла hello.txt и добавление его в локальный репозиторий:

```
echo ‘hello world’ > hello.txt
git add hello.txt
git commit -am ‘Новый файл’
```

9. Ветки нужны для того, чтобы при работе над проектом программисты работали незваисимо друг от друга. В Git ветки — это элемент повседневного процесса разработки. По сути ветки в Git представляют собой указатель на снимок изменений. Если нужно добавить новую возможность или исправить ошибку (незначительную или серьезную), создается новая ветка, в которой будут размещаться эти изменения.

10. Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты. Игнорируемые файлы отслеживаются в специальном файле .gitignore, который регистрируется в корневом каталоге репозитория. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо

вручную отредактировать файл .gitignore, чтобы указать в нем новые файлы, которые должны быть проигнорированы. Файлы .gitignore содержат шаблоны, которые сопоставляются с именами файлов в репозитории для определения необходимости игнорировать эти файлы.

5 Вывод

В ходе выполнения данной лабораторной работы я повторила процесс оформления отчётов с помощью легковесного языка разметки Markdown.

Список литературы

1. Markdown [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Markdown.