МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 2

по дисциплине

«Технология разработки программных приложений»

Выполнил:
Студент группы ИКБО-10-23 Лазаренко С. А.

Проверил:
Преподаватель к.э.н., доцент Петросян Л. Э.

Москва 2025 г.

СОДЕРЖАНИЕ

ЧАСТЬ 1. ОСНОВНЫЕ КОМАНДЫ Git

1. Установка и настройка клиент git на рабочей станции

Сначала зайдем на официальный сайт Git:

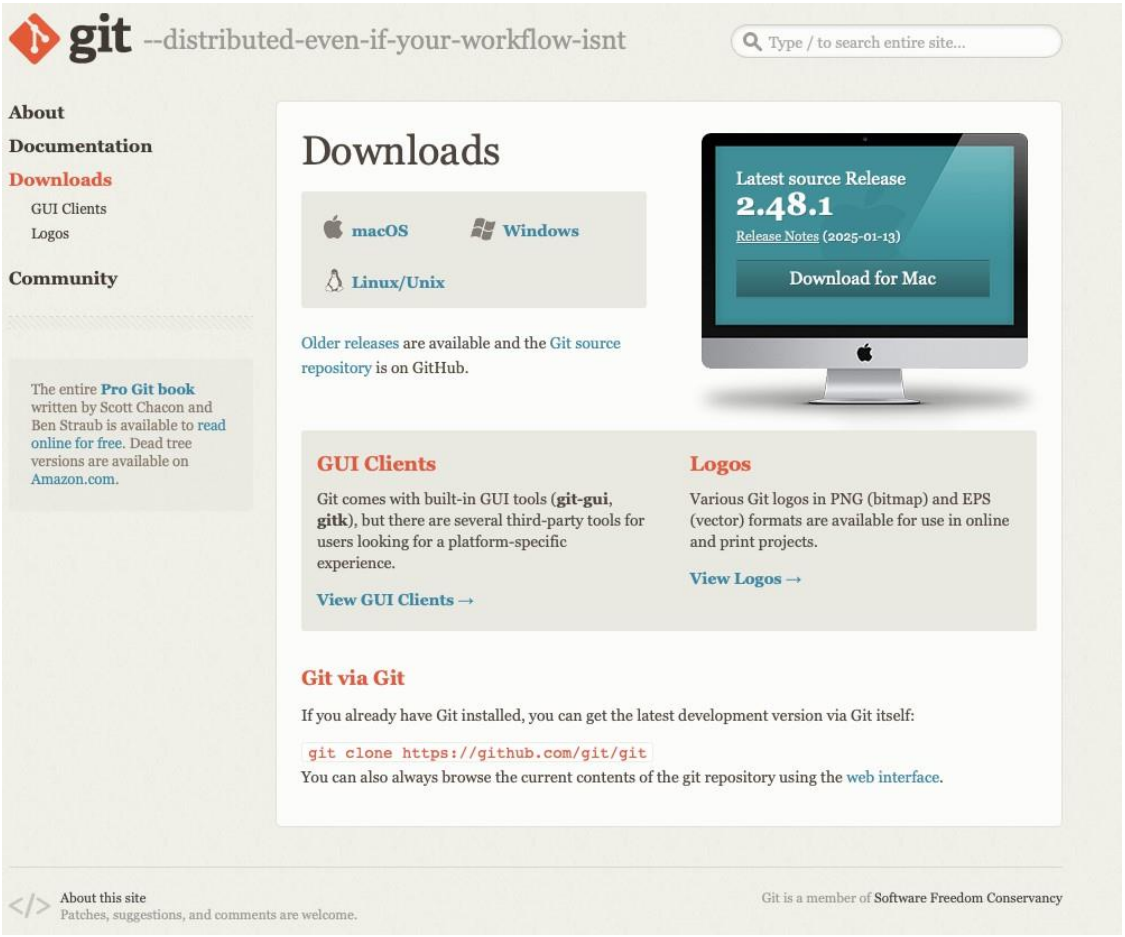

Рисунок 1.1 – Главная страница сайта Git

Далее выберем нужную версию под операционную систему (macOS) и скачаем установочный файл. После этого запустим его и , следуя инструкциям на экране, установим Git.

После завершения установки откроем терминал и проверим, что система появилась на устройстве:

```
gwynbleidd@MacBook-Air-Sergej-3 ~ % git --version
git version 2.46.0
```

Рисунок 1.2 – Проверка версии Git в консоли

Откроем терминал и выполним следующие команды в консоле:

```
gwynbleidd@MacBook-Air-Sergej-3 ~ % git config --global user.name "Sergey Lazarenko"
git config --global user.email "sergey.lazarenko.0241@mail.ru"
```

Рисунок 1.3 – Настройка имени пользователя и электронной почты в Git

Выполним следующую команду и для проверки верности всех введенных команд, напишем *git config --list*:

```
gwynbleidd@MacBook-Air-Sergej-3 ~ % git config --global core.quotepath off
```

Рисунок 1.4 – Отключение кавычек для путей файлов в Git

```
gwynbleidd@MacBook-Air-Sergej-3 ~ % git config --list
credential.helper=osxkeychain
user.name=Sergey Lazarenko
user.email=sergey.lazarenko.0241@mail.ru
core.excludesfile=~/.gitignore_global
core.quotepath=off
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=true
```

Рисунок 1.5 – Базовая настройка Git для macOS

2. Создание локального репозитория и добавления в него файлов

Сначала инициализируем локальный репозиторий:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Инициализирован пустой репозиторий Git в /Users/gwynbleidd/Desktop/prac1/.git/
```

Рисунок 1.6 – Инициализация пустого репозитория Git

Создадим несколько файлов, затем добавим их в репозиторий:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % touch file1.txt file2.txt file3.txt

gwynbleidd@MacBook-Air-Sergej-3 prac1 % git add file1.txt file2.txt file3.txt

gwynbleidd@MacBook-Air-Sergej-3 prac1 % git commit -m "Добавлены файлы file1.txt, file2.txt, file3.txt"

[master 645ec75] Добавлены файлы file1.txt, file2.txt, file3.txt
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.txt
 create mode 100644 file2.txt
 create mode 100644 file3.txt
```

Рисунок 1.7 – Создание и добавление файлов в локальный репозиторий

3. Внесение изменений в один из файлов

Откроем файл с помощью текстового редактора nano для внесения изменений в один из файлов:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % nano file1.txt
```

Рисунок 1.8 – Внесение изменений в файле

4. Индексация изменений и проверка состояния

Внесем изменений в созданный ранее файл и проверим его статус с помощью команды *git status*:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % nano prac1.htmly
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git status
Текущая ветка: master
Изменения, которые не в индексе для коммита:
  (используйте «git add <файл>...», чтобы добавить файл в индекс)
  (используйте «git restore <файл>...», чтобы отменить изменения в рабочем каталоге)
        изменено:      prac1.htmly

индекс пуст (используйте «git add» и/или «git commit -a»)
```

Рисунок 1.9 – Индексация изменений

5. Коммит того, что было проиндексировано. Добавления комментария

Убедимся, что файл добавлен в индекс, затем добавим коммит с комметарием:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % nano file1.txt
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git add file1.txt
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git status
Текущая ветка: master
Изменения, которые будут включены в коммит:
  (используйте «git restore --staged <файл>...», чтобы убрать из индекса)
        изменено:      file1.txt

gwynbleidd@MacBook-Air-Sergej-3 prac1 % git commit -m "BIba"
[master 8c486a3] BIba
 1 file changed, 1 insertion(+)
```

Рисунок 1.10 – Коммит изменений с комментарием в Git’

6. Коммиты нескольких изменений

Откроем файл с помощью текстового редактора nano, чтобы внести изменения в файл prac1.html. После того как внесены изменения, выполним команды для добавления и коммита изменений в Git для фиксации изменений:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % nano prac1.html
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git add prac1.html
gwynbleidd@MacBook-Air-Sergej-3 prac1 % nano prac1.html
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git status prac1.html
Текущая ветка: master
Изменения, которые будут включены в коммит:
  (используйте «git restore --staged <файл>...», чтобы убрать из индекса)
        изменено:      prac1.html

Изменения, которые не в индексе для коммита:
  (используйте «git add <файл>...», чтобы добавить файл в индекс)
  (используйте «git restore <файл>...», чтобы отменить изменения в рабочем каталоге)
        изменено:      prac1.html

gwynbleidd@MacBook-Air-Sergej-3 prac1 % git commit -m "Биба боба"
[master 6f63f75] Биба боба
 1 file changed, 1 insertion(+), 1 deletion(-)
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git status
Текущая ветка: master
Изменения, которые не в индексе для коммита:
  (используйте «git add <файл>...», чтобы добавить файл в индекс)
  (используйте «git restore <файл>...», чтобы отменить изменения в рабочем каталоге)
        изменено:      prac1.html
        изменено:      prac1.htmly

индекс пуст (используйте «git add» и/или «git commit -a»)
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git add .
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git status
Текущая ветка: master
Изменения, которые будут включены в коммит:
  (используйте «git restore --staged <файл>...», чтобы убрать из индекса)
        изменено:      prac1.html
        изменено:      prac1.htmly

gwynbleidd@MacBook-Air-Sergej-3 prac1 % git commit -m "Бом Бом"
[master 060273a] Бом Бом
 2 files changed, 2 insertions(+), 2 deletions(-)
```

Рисунок 1.11 – Коммиты нескольких изменений

7. Просмотр истории коммитов

Для просмотра истории коммитов воспользуемся командой *git log*:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git log
commit 060273a09890eadff844354bedf6f2e0434a2cb7 (HEAD -> master)
Author: Sergey Lazarenko <sergey.lazarenko.0241@mail.ru>
Date:   Thu Feb 13 23:06:46 2025 +0300

    Бом Бом

commit 6f63f757037791d58d8adb7fed0ec0d439466799
Author: Sergey Lazarenko <sergey.lazarenko.0241@mail.ru>
Date:   Thu Feb 13 23:06:06 2025 +0300

    Биба боба

commit c1a90289060bf0dd4d87bb4d717ebb2dc7f45377
Author: Sergey Lazarenko <sergey.lazarenko.0241@mail.ru>
Date:   Thu Feb 13 22:52:17 2025 +0300

    Initial commit

commit 811356fabe8d0c1e02f0aade3671ca0b455a924a
Author: Sergey Lazarenko <sergey.lazarenko.0241@mail.ru>
Date:   Thu Feb 13 22:51:30 2025 +0300

    Initial commit
```

Рисунок 1.12 – Просмотр истории коммитов

Команда *git log* позволяет контролировать формат выводимой информации:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git log --pretty=oneline
060273a09890eadff844354bedf6f2e0434a2cb7 (HEAD -> master) Бом Бом
6f63f757037791d58d8adb7fed0ec0d439466799 Биба боба
c1a90289060bf0dd4d87bb4d717ebb2dc7f45377 Initial commit
811356fabe8d0c1e02f0aade3671ca0b455a924a Initial commit
```

Рисунок 1.13 – Изменение формата вывода коммитов

8. Получение старых версий рабочего каталога

Для получения старой версии необходимо узнать для начала хэши, для этого воспользуемся командой из предыдущего шага:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short
* 060273a 2025-02-13 | Бом Бом (HEAD -> master) [Sergey Lazarenko]
* 6f63f75 2025-02-13 | Биба боба [Sergey Lazarenko]
* c1a9028 2025-02-13 | Initial commit [Sergey Lazarenko]
* 811356f 2025-02-13 | Initial commit [Sergey Lazarenko]
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git checkout  811356f
Примечание: переключение на «811356f».

Вы сейчас в состоянии «отсоединённого указателя HEAD». Можете осмотреться,
внести экспериментальные изменения и зафиксировать их, также можете
отменить любые коммиты, созданные в этом состоянии, не затрагивая другие
ветки, переключившись обратно на любую ветку.

Если хотите создать новую ветку для сохранения созданных коммитов, можете
сделать это (сейчас или позже), используя команду switch с параметром -c.
Например:

  git switch -c <новая-ветка>

Или отмените эту операцию с помощью:

  git switch -

Отключите этот совет, установив переменную конфигурации
advice.detachedHead в значение false

HEAD сейчас на 811356f Initial commit
gwynbleidd@MacBook-Air-Sergej-3 prac1 % cat prac1.htmly
cat: prac1.htmly: No such file or directory
gwynbleidd@MacBook-Air-Sergej-3 prac1 % cat prac1.html
Описание для души с вечерним вайбом
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git checkout master
Предыдущая позиция HEAD была 811356f Initial commit
Переключились на ветку «master»
gwynbleidd@MacBook-Air-Sergej-3 prac1 % cat prac1.html
12312asdasdОписание для души с вечерним вайбом
```

Рисунок 1.14 – Получение старых версий

9. Изучение создания тегов для коммитов для использования в будущем Создадим легкий тег – ссылка на коммит, который не содержит дополнительной информации, просмотрим список всех тегов, создадим тег для конкретного коммита, а не для последнего, указав его хеш:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git tag v1.0
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git tag
v1.0
```

Рисунок 1.15 – Создание и просмотр тега

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git tag v1.1 6ee897aa2ecc665b1f955da59dce67f6dbdd4541
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git tag
v1.0
v1.1
```

Рисунок 1.16 – Создание тега для конкретного коммита с указанием хеша

10. Отмена локальных изменений (до индексации и после индексации)

Проверим статус репозитория с помощью *git status*, увидим, что файл был изменен, но не добавлен в индексации для коммита. Затем используем команду *git checkout prac1.html*, чтобы отменить изменения и вернуть файл в состояние последнего коммита, после чего проверим содержимое с помощью *cat*:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git checkout master
Уже на «master»
gwynbleidd@MacBook-Air-Sergej-3 prac1 % nano prac1.html
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git status
Текущая ветка: master
Изменения, которые не в индексе для коммита:
  (используйте «git add <файл>...», чтобы добавить файл в индекс)
  (используйте «git restore <файл>...», чтобы отменить изменения в рабочем каталоге)
        изменено:      prac1.html

индекс пуст (используйте «git add» и/или «git commit -a»)
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git checkout prac1.html
Updated 1 path from the index
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git status
Текущая ветка: master
нечего коммитить, нет изменений в рабочем каталоге
gwynbleidd@MacBook-Air-Sergej-3 prac1 % cat prac1.html
12312asdasdОписание для души с вечерним вайбом
```

Рисунок 1.17 – Отмена локальных изменений до индексации

Внесем изменения в файл и добавим его в индекс с помощью команды *git add*. Затем, командой *git reset HEAD*, отменим добавление файла в индекс, вернув его состояние “неидексированных изменений”. После используем команду *git checkout prac1.html*, чтобы отменить все изменения в файле и восстановить последную сохраненную версию:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % nano prac1.html
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git add prac1.html
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git status
Текущая ветка: master
Изменения, которые будут включены в коммит:
  (используйте «git restore --staged <файл>...», чтобы убрать из индекса)
        изменено:      prac1.html

gwynbleidd@MacBook-Air-Sergej-3 prac1 % git reset HEAD prac1.html
Непроиндексированные изменения после сброса:
M       prac1.html
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git checkout prac1.html
Updated 1 path from the index
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git status
Текущая ветка: master
нечего коммитить, нет изменений в рабочем каталоге
```

Рисунок 1.18 – Отмена локальных изменений после индексации, до коммита

11. Отмена коммита в локальном репозитории

Внесем изменения в файл, добавим его в индекс, а затем зафиксируем изменения с помощью команды *git commit*. После этого настроим редактор по умолчанию для Git на nano. Затем отменим последний коммит, создав новый коммит, который отменяет изменения старого. Проверим историю коммитов, убедившись в присутствии коммита отмены:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % nano prac1.html
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git add prac1.html
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git commit -m "Love666"
[master 288b342] Love666
 1 file changed, 1 insertion(+), 1 deletion(-)
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git config --global core.editor nano
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git revert HEAD
[master b686af1] Revert "Commit to be removed"
 1 file changed, 1 insertion(+), 1 deletion(-)
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short
* b686af1 2025-02-14 | Revert "Commit to be removed" (HEAD -> master) [Sergey Lazarenko]
* 288b342 2025-02-14 | Love666 [Sergey Lazarenko]
* 6ee897a 2025-02-14 | Love666 [Sergey Lazarenko]
* 060273a 2025-02-13 | Бом Бом [Sergey Lazarenko]
* 6f63f75 2025-02-13 | Биба боба [Sergey Lazarenko]
* c1a9028 2025-02-13 | Initial commit [Sergey Lazarenko]
* 811356f 2025-02-13 | Initial commit [Sergey Lazarenko]
```

Рисунок 1.19 – Отмена коммита

ЧАСТЬ 2. СИСТЕМЫ УПРАВЛЕНИЯ РЕПОЗИТОРИЯМИ

1. Создание аккаунта на GitHub

Перейдем на сайт GitHub, пройдем регистрацию, после которой будет небольшой опрос.

После прохождения всех этапов на сайте, на указанный при регистрации ящик придет письмо от GitHub. Отроем его и подтвердим свой почтовый адрес. Теперь у нас есть профиль на GitHub:

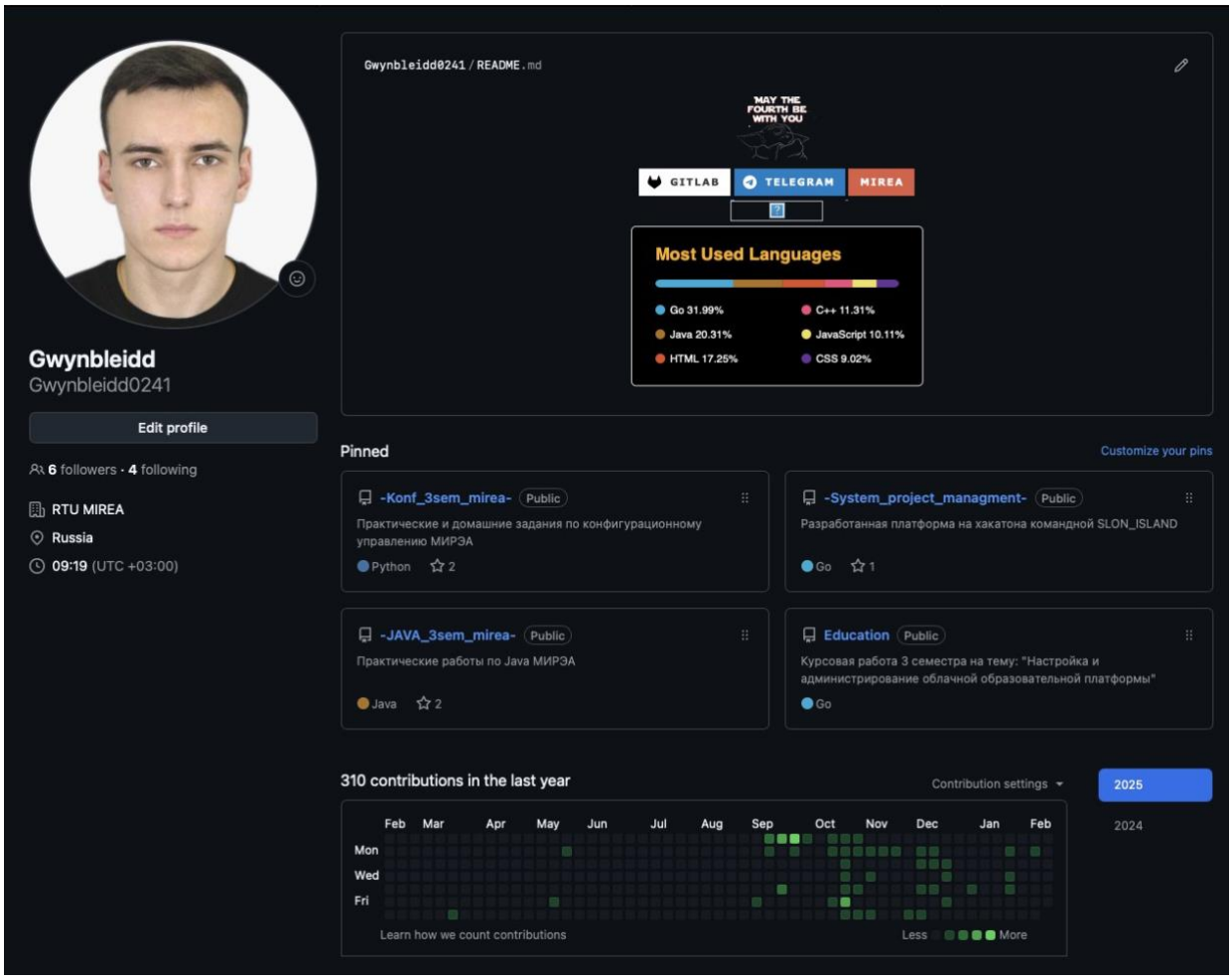

Рисунок 2.1 – Созданный профиль на GitHub

2. Создание SSH-ключа для авторизации

Сначала проверим, есть ли уже на компьютере ключ. По умолчанию SSH-ключи храняться в каталоге *~/.ssh*, поэтому нужно проверить содержимое этого каталога:

```
gwynbleidd@MacBook-Air-Sergej-3 ~ % cd ~/.ssh
gwynbleidd@MacBook-Air-Sergej-3 .ssh % ls
```

Рисунок 2.2 – Проверка каталога, содержащего SSH-ключи

Открываем консоль и вводим команду:

```
gwynbleidd@MacBook-Air-Sergej-3 .ssh % ssh-keygen -t rsa -b 4096 -C "sergey.lazarenko.0241@mail.ru"
```

Рисунок 2.3 – Создание SSH-ключа

Далее добавим ключ в ssh-agent (сгенерированный или уже существующий) с помощью *ssh-add ~/.ssh/id_ed25519.pub*. В результате получим:

```
SSH_AUTH_SOCK=/var/folders/7_/hh7_wr1s33dbjqdyq4czdz140000gn/T//ssh-yktDdYxkHVsk/agent.62437; export SSH_AUTH_SOCK;
SSH_AGENT_PID=62438; export SSH_AGENT_PID;
echo Agent pid 62438;
```

Рисунок 2.4 – Настройка SSH-агента и проверка его процесса

Чтобы связать локальный и удаленный репозитории, выведем в консоль содержимое файла с помощью *cat*. В итоге получим вот такой ключ (с методички):


```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDOR6Mi4vahmvV2hF7A38itIyhEpKxZzVHt8YWHxrIhraIkWJzh3chDr6p4AFYcN3rtCtDi1E4GekI9DdVt7NwmPh
OyCsVa7330K+rHoRHxYiZouk/1+qXpr92thUGuViSyHUNO0nGIgY8WO7XmXVczhBfudL3OvwN4yW5FWze/c4dGVlY7XH5HVrfb+Px5FITS99yPIS1ZjCpk0d3L2L5d
Uiv+TzTyTPe2sR1zS69Db1XSvrLtokKeOriDwsLC7/xnjtJOAEGnwbrqXFM5pSMYTZJtXyGwgQ27l5aXVizmjuY9fDiH5PQscoA9aevmDUIgeEQFx4YIwHoXnKKwmS
YvP+9NRtUiHv1S7COLpuh715ZXDCaW16XCQjfMioa/A4/pVOdS1D7YkcXhmDRzw8Gh5x99D/AJ0VDBTezAi9ejbqaYaBsxI280El++2kGI+k/vx3jgaBK1xFfuwXc/
Ya76GmB9S0gzK9pIi52MIa4hgTv6Z+gerW69o6N59f8qUPl5ANYkMtbNfjNbEAZRQ4MAp1qr+R1dvc8IZC/pKgIWg4nAIl89BGm+Mfj9SMUDSwIt0sjOGXHNn+Y+jh
```


Рисунок 2.5 – Полученный SSH-ключ

Скопируем его из консоли и перейдем на страницу для работы с ключами в нашем профиле на GitHub. Выбирем кнопку “New SSH key”, откроется окно с вводом данных, в поле “key” вставим скопированный ключ, в “Title” вводим любое имя ключа и нажимаем “Add SSH key”:

Рисунок 2.6 – Добавленный SSH-ключ в GitHub

3. Создание нового репозитория для своего проекта

Создадим новую директорию, затем создадим несколько файлов в этой директории с добавлением содержимого в файлы. После инициализируем новый Git-репозиторий:

```
gwynbleidd@MacBook-Air-Sergej-3 Desktop % mkdir Pr1
gwynbleidd@MacBook-Air-Sergej-3 Desktop % cd Pr1
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % touch readme.txt
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % nano readme.txt
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % touch helloworld.go & goodbyewordl.go
[1] 26568
zsh: command not found: goodbyewordl.go
gwynbleidd@MacBook-Air-Sergej-3 Pr1 %
[1]  + done       touch helloworld.go
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % touch goodbyeword.go
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % nano goodbyeword.go
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Инициализирован пустой репозиторий Git в /Users/gwynbleidd/Desktop/Pr1/.git/
```

Рисунок 2.7 – Создание репозитория с несколькими файлами

4. Связывание локального и удаленного репозитория

Заходим на GitHub, на свою страницу, выбираем вкладку репозитория. На вкладке с репозиториями нажимаем на зеленую кнопку “New”. Называем наш репозиторий и выбираем приватность репозитория. После проделанных шагов, мы создали пустой репозиторий на GitHub, осталось только связать его с локальным:

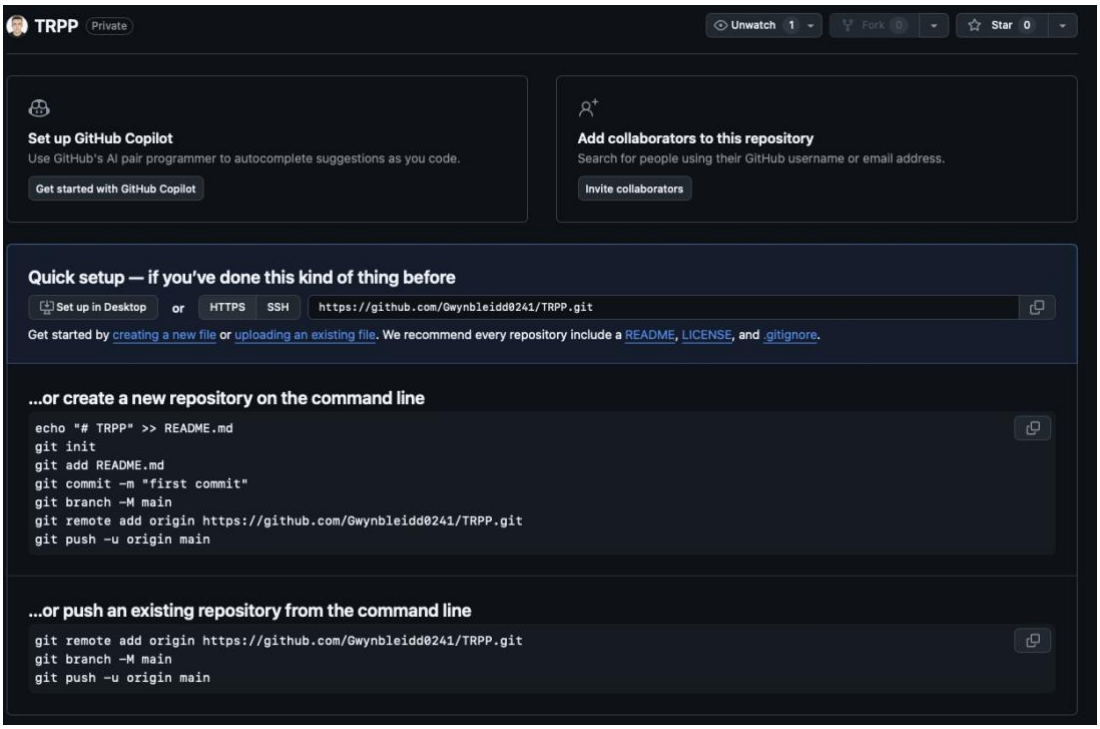

Рисунок 2.8 – Созданный репозиторий на GitHub

Чтобы связать репозитории друг с другом введем следующую команду в консоль:

```
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git remote add Pr1 git@github.com:Gwynbleidd0241/TRPP.git
```

Рисунок 2.9 – Связывание локального и удаленного репозитория

5. Создание веток и переключение между ними

Для создания новой ветки и переключения на нее воспользуемся командой *git checkout -b <имя ветки>*. Сделаем изменения в файле и проиндексируем их, затем коммитим:

```
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git checkout -b dev
Переключились на новую ветку «dev»
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % touch 123.py && nano 123.py
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git add 123.py
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git status
Текущая ветка: dev

Еще нет коммитов

Изменения, которые будут включены в коммит:
  (используйте «git rm --cached <файл>...», чтобы убрать из индекса)
        новый файл:    123.py

Неотслеживаемые файлы:
  (используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)
        goodbyeword.go
        helloworld.go
        readme.txt

gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git commit -m "Initial commit"
[dev (корневой коммит) da9cfc7] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 123.py
```

Рисунок 2.10 – Работа в новой ветке

6. Слияние веток

Воспользуемся командой *git merge* для слияния веток dev и master:

```
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git checkout master
D       123.py
Переключились на ветку «master»
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git merge dev
```

Рисунок 2.11 – Слияние веток

7. Выполнение цепочки действий в репозитории согласно варианту 6

Клонируем непустой удаленный репозиторий на локальную машину:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git clone git@github.com:Gwynbleidd0241/-System_project_managment-.git
Клонирование в «-System_project_managment-»...
Enter passphrase for key '/Users/gwynbleidd/.ssh/id_ed25519':
remote: Enumerating objects: 500, done.
remote: Counting objects: 100% (500/500), done.
remote: Compressing objects: 100% (270/270), done.
remote: Total 500 (delta 113), reused 495 (delta 111), pack-reused 0 (from 0)
Получение объектов: 100% (500/500), 18.87 МиБ | 4.27 МиБ/с, готово.
Определение изменений: 100% (113/113), готово.
```

Рисунок 2.12 – Клонирование непустого удаленного репозитория

Создадим новую ветку и выведим список всех веток:

```
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git checkout -b dev
Переключились на новую ветку «dev»
gwynbleidd@MacBook-Air-Sergej-3 prac1 % git branch -a
* dev
  master
```

Рисунок 2.13 – Создание новой ветки и вывод списка всех веток

Проведем 3 коммита в новой ветке в разные файлы:

```
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % touch file1.txt file2.txt file3.txt
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git add file1.txt
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git commit -m "file1.txt"
[master 1ccbae8] file1.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.txt
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git add file2.txt
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git commit -m "file2.txt"
[master 5ac2b5b] file2.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file2.txt
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git add file3.txt
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git commit -m "file3.txt"
[master b9a65ff] file3.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file3.txt
```

Рисунок 2.14 – Произведение 3 коммитов в новой ветке

Выгрузим изменения в удаленный репозиторий:

```
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git push origin dev
fatal: 'origin' does not appear to be a git repository
fatal: Не удалось прочитать из внешнего репозитория.

Удостоверьтесь, что у вас есть необходимые права доступа
и репозиторий существует.
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git remote add origin https://github.com/Gwynbleidd0241/TRPP.git

gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git push origin dev
Перечисление объектов: 10, готово.
Подсчет объектов: 100% (10/10), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (7/7), готово.
Запись объектов: 100% (10/10), 839 байтов | 839.00 КиБ/с, готово.
Total 10 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Gwynbleidd0241/TRPP.git
 * [new branch]      dev -> dev
```

Рисунок 2.15 – Выгрузка изменений в удаленный репозиторий

Переключимся на ветку master, затем с помощью nano изменим файл, не производя коммит:

```
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git checkout master
D       123.py
Переключились на ветку «master»
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % nano file1.txt
```

Рисунок 2.16 – Изменения в файле в ветке master

Выполним команду *git stash*, которая сохранит изменения в хранилище и вернет рабочую директорию до состояния последнего коммита, затем переключимся на ветку dev и выполним *git stash pop*, которая восстановит изменения в текущей ветке и удалит их из хранилища:

```
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git stash
Рабочий каталог и состояние индекса сохранены WIP on master: b9a65ff file3.txt
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git checkout
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git checkout deev
error: pathspec 'deev' did not match any file(s) known to git
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git checkout dev
Переключились на ветку «dev»
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git stash pop
Текущая ветка: dev
Изменения, которые не в индексе для коммита:
  (используйте «git add/rm <файл>...», чтобы добавить или удалить файл из индекса)
  (используйте «git restore <файл>...», чтобы отменить изменения в рабочем каталоге)
        удалено:       123.py
        изменено:      file1.txt

Неотслеживаемые файлы:
  (используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)
        goodbyeword.go
        helloworld.go
        readme.txt

индекс пуст (используйте «git add» и/или «git commit -a»)
Отброшено refs/stash@{0} (6ceea3b20b65b21d72b994ede325bbf00b22c8de)
```

Рисунок 2.17 – Откат изменений в новой ветке

Выведим в консоли различия между веткой master и dev, затем сольем ветки при помощи *git merge*:

```
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git diff dev..master

gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git diff master..dev
```

Рисунок 2.18 – Различия в ветках

```
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git checkout master
M       123.py
M       file1.txt
Переключились на ветку «master»
gwynbleidd@MacBook-Air-Sergej-3 Pr1 % git merge dev
Уже актуально.
```

Рисунок 2.19 – Слияние веток

ЧАСТЬ 3. РАБОТА С ВЕТВЛЕНИЕМ И ОФОРМЛЕНИЕ КОДА

1. Получение форка репозитория в соответствии с вариантом 7

Заходим на GitHub, переходим по ссылке в репозиторий, который собираемся форкать, в верхнем правом углу нажимает кнопку “Fork”:

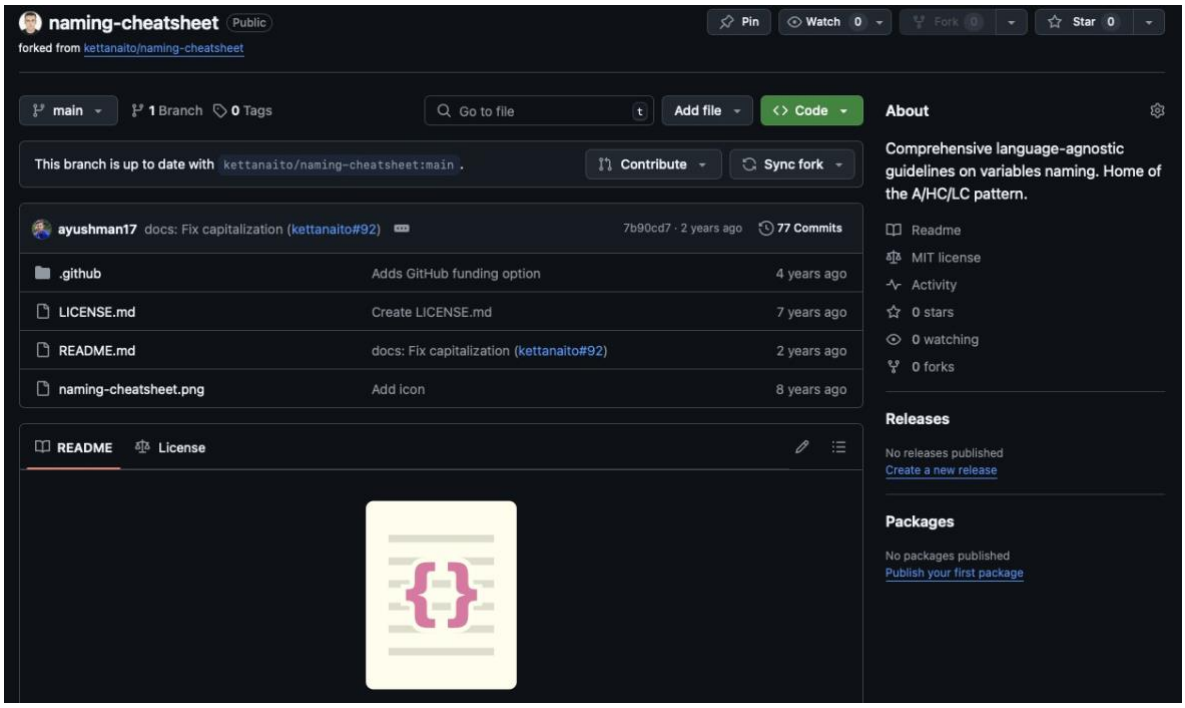

Рисунок 3.1 – Форк репозитория

2. Клонирование репозитория на локальную машину

```
gwynbleidd@MacBook-Air-Sergej-3 Desktop % git clone https://github.com/Gwynbleidd0241/naming-cheatsheet.git
Клонирование в «naming-cheatsheet»...
remote: Enumerating objects: 214, done.
remote: Counting objects: 100% (68/68), done.
remote: Compressing objects: 100% (38/38), done.
remote: Total 214 (delta 40), reused 30 (delta 30), pack-reused 146 (from 1)
Получение объектов: 100% (214/214), 63.66 КиБ | 749.00 КиБ/с, готово.
Определение изменений: 100% (78/78), готово.
```

Рисунок 3.2 – Клонирование репозитория на локальную машину

3. Создание двух веток в main’е

Проверим, что мы находимся на нужной ветке и создадим две ветки от последнего коммита в ветке main:

```
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git checkout main
Уже на «main»
Эта ветка соответствует «origin/main».
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git checkout -b branch1
Переключились на новую ветку «branch1»
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git checkout main
Переключились на ветку «main»
Эта ветка соответствует «origin/main».
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git checkout -b branch2
Переключились на новую ветку «branch2»
```

Рисунок 3.3 – Создание веток от последнего коммита в main

4. Проведение по 3 коммита в каждую из веток

Переключимся на branch1, проведем по 3 коммита, затем также на ветку branch2:

```
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git checkout branch1

Переключились на ветку «branch1»
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % echo "Первое изменение в branch1" > file.txt
git add file.txt
git commit -m "Первое изменение в branch1"

[branch1 e0a8525] Первое изменение в branch1
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % echo "Второе изменение в branch1" >> file.txt
git add file.txt
git commit -m "Второе изменение в branch1"

[branch1 c05c1a5] Второе изменение в branch1
 1 file changed, 1 insertion(+)
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % echo "Третье изменение в branch1" >> file.txt
git add file.txt
git commit -m "Третье изменение в branch1"

[branch1 362c623] Третье изменение в branch1
 1 file changed, 1 insertion(+)
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git checkout branch2

Переключились на ветку «branch2»
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % echo "Первое изменение в branch2" > file.txt
git add file.txt
git commit -m "Первое изменение в branch2"

[branch2 df2d1fc] Первое изменение в branch2
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % echo "Второе изменение в branch2" >> file.txt
git add file.txt
git commit -m "Второе изменение в branch2"

[branch2 e7bed1b] Второе изменение в branch2
 1 file changed, 1 insertion(+)
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % echo "Третье изменение в branch2" >> file.txt
git add file.txt
git commit -m "Третье изменение в branch2"

[branch2 e4ba5b0] Третье изменение в branch2
 1 file changed, 1 insertion(+)
```

Рисунок 3.4 – Проведение по 3 коммита в каждую из веток

5. Слияние ветки branch1 в ветку branch2

Выполним слияние ветки branch1 в ветку branch2, разрешив конфликты при этом:

```
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git checkout branch2

Уже на «branch2»
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git merge branch1
Автослияние file.txt
КОНФЛИКТ (добавление/добавление): Конфликт слияния в file.txt
Сбой автоматического слияния; исправьте конфликты, затем зафиксируйте результат.
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % nano file.txt
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git merge branch1
error: Невозможно выполнить слияние, так как у вас имеются не слитые файлы.
hint: Исправьте их в рабочем каталоге, затем запустите «git add/rm <файл>»,
hint: чтобы пометить исправление и сделайте коммит.
fatal: Выход из-за неразрешенного конфликта.
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git add file.txt

gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git commit -m "Разрешены конфликты при слиянии branch1 с branch2"

[branch2 ed8e827] Разрешены конфликты при слиянии branch1 с branch2
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git merge branch1
Уже актуально.
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet %
```

Рисунок 3.5 – Слияние веток

6. Выгрузка всех изменений во всех ветках в удаленный репозиторий

```
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git remote add origin git@github.com:Gwynbleidd0241/naming-cheatsheet.git

error: внешний репозиторий origin уже существует
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git push --all origin

Перечисление объектов: 22, готово.
Подсчет объектов: 100% (22/22), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (19/19), готово.
Запись объектов: 100% (21/21), 2.11 КиБ | 2.11 МиБ/с, готово.
Total 21 (delta 11), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (11/11), completed with 1 local object.
To https://github.com/Gwynbleidd0241/naming-cheatsheet.git
 * [new branch]      branch1 -> branch1
 * [new branch]      branch2 -> branch2
```

Рисунок 3.6 – Выгрузка всех изменений

7. Проведение еще 3 коммитов в ветке branch1

```
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git checkout branch1

Переключились на ветку «branch1»
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % echo "Первое изменение в branch1" >> file.txt
git add file.txt
git commit -m "Первое изменение в branch1"

[branch1 0c8e629] Первое изменение в branch1
 1 file changed, 1 insertion(+)
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % echo "Второе изменение в branch1" >> file.txt
git add file.txt
git commit -m "Второе изменение в branch1"

[branch1 0ea8f51] Второе изменение в branch1
 1 file changed, 1 insertion(+)
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % echo "Третье изменение в branch1" >> file.txt
git add file.txt
git commit -m "Третье изменение в branch1"

[branch1 53396b9] Третье изменение в branch1
 1 file changed, 1 insertion(+)
```

Рисунок 3.7 – Проведение 3 коммитов в ветке branch1

8. Клонирование репозитория в другую директорию

```
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % cd ~
gwynbleidd@MacBook-Air-Sergej-3 ~ % git clone git@github.com:Gwynbleidd0241/naming-cheatsheet.git clone2
Клонирование в «clone2»...
Enter passphrase for key '/Users/gwynbleidd/.ssh/id_ed25519':
remote: Enumerating objects: 235, done.
remote: Counting objects: 100% (89/89), done.
remote: Compressing objects: 100% (46/46), done.
remote: Total 235 (delta 51), reused 51 (delta 41), pack-reused 146 (from 1)
Получение объектов: 100% (235/235), 65.62 КиБ | 646.00 КиБ/с, готово.
Определение изменений: 100% (89/89), готово.
```

Рисунок 3.8 – Клонирование репозитория в другую директорию

9. Проведение 3 коммитов в ветке branch1 в новом клоне репозитория

```
gwynbleidd@MacBook-Air-Sergej-3 ~ % cd clone2
gwynbleidd@MacBook-Air-Sergej-3 clone2 % git checkout branch1
branch 'branch1' set up to track 'origin/branch1'.
Переключились на новую ветку «branch1»
gwynbleidd@MacBook-Air-Sergej-3 clone2 % echo "Первое изменение в branch1" >> file.txt
git add file.txt
git commit -m "Первое изменение в branch1"

[branch1 fc605b1] Первое изменение в branch1
 1 file changed, 1 insertion(+)
gwynbleidd@MacBook-Air-Sergej-3 clone2 % echo "Второе изменение в branch1" >> file.txt
git add file.txt
git commit -m "Второе изменение в branch1"

[branch1 b307be2] Второе изменение в branch1
 1 file changed, 1 insertion(+)
gwynbleidd@MacBook-Air-Sergej-3 clone2 % echo "Третье изменение в branch1" >> file.txt
git add file.txt
git commit -m "Третье изменение в branch1"

[branch1 48afd2f] Третье изменение в branch1
 1 file changed, 1 insertion(+)
```

Рисунок 3.9 – Проведение 3 коммитов в новом клоне репозитортия

10. Выгрузить все изменения из нового репозитория в удаленный

```
gwynbleidd@MacBook-Air-Sergej-3 clone2 % git add .

gwynbleidd@MacBook-Air-Sergej-3 clone2 % git commit -m "измененiz"

Текущая ветка: branch1
Ваша ветка опережает «origin/branch1» на 3 коммита.
  (используйте «git push», чтобы опубликовать ваши локальные коммиты)

нечего коммитить, нет изменений в рабочем каталоге
gwynbleidd@MacBook-Air-Sergej-3 clone2 % git push --all origin

Enter passphrase for key '/Users/gwynbleidd/.ssh/id_ed25519':
Перечисление объектов: 11, готово.
Подсчет объектов: 100% (11/11), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (9/9), готово.
Запись объектов: 100% (9/9), 930 байтов | 930.00 КиБ/с, готово.
Total 9 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To github.com:Gwynbleidd0241/naming-cheatsheet.git
   362c623..48afd2f  branch1 -> branch1
```

Рисунок 3.10 – Выгрузка всех изменений в удаленный репозиторий

11. Возвращение в старый клон с репозиторием, выгрузка изменений

```
gwynbleidd@MacBook-Air-Sergej-3 Desktop % cd naming-cheatsheet
gwynbleidd@MacBook-Air-Sergej-3 naming-cheatsheet % git push --all --force origin

Перечисление объектов: 11, готово.
Подсчет объектов: 100% (11/11), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (9/9), готово.
Запись объектов: 100% (9/9), 931 байт | 465.00 КиБ/с, готово.
Total 9 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To https://github.com/Gwynbleidd0241/naming-cheatsheet.git
 + 48afd2f...53396b9 branch1 -> branch1 (forced update)
```

Рисунок 3.11 – Возвращение в старый клон и выгрузка изменений

12. Получение всех изменений в новом репозитории

```
gwynbleidd@MacBook-Air-Sergej-3 clone2 % git fetch origin
Enter passphrase for key '/Users/gwynbleidd/.ssh/id_ed25519':
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Распаковка объектов: 100% (3/3), 619 байтов | 206.00 КиБ/с, готово.
Из github.com:Gwynbleidd0241/naming-cheatsheet
 + 48afd2f...53396b9 branch1    -> origin/branch1  (принудительное обновление)
```

Рисунок 3.12 – Получение изменений

Контрольные вопросы

1. Как отменить слияние веток, если произошел конфликт?

Отменить слияние веток при появлении конфликта можно командой *git merge --abort.*

2. Для чего нужен .gitignore?

.gitignore используется для указания файлов и директорий, которые не должны учитываться системой контроля версий Git. Это позволяет игнорировать временные файлы, конфигурационные файлы, и прочее, чтобы они не попадали в репозиторий.

3. Что делает команда git status?

Команда *git status* показывает текущий статус репозитория, включая измененные, добавленные и неотслеживаемые файлы.

4. Что делает команда git add?

Команда *git add* добавляет изменения файлов в индекс, подготавливая их к следующему коммиту.

5. Что делает команда git log?

Команда *git log* показывает историю коммитов. Она перечисляет коммиты, которые можно достичь, следуя связям родительских коммитов от указанных коммитов.

6. Что делает команда git diff?

Команда *git diff* показывает различия между рабочим деревом и индексом, а также между индексом и последним коммитом.

7. Что делает команда git show?

Команда *git show* отображает информацию о конкретном коммите, включая изменения, внесенные этим коммитом.

8. Что делает команда git stash?

Команда *git stash* временно сохраняет изменения в рабочей директории, чтобы можно было переключиться на другую ветку или выполнить другие операции.

ВЫВОД

Мы освоили основные команды Git для управления локальным репозиторием: создание, изменение, индексирование, коммиты и отмена изменений. Также изучили работу с тегами для метки коммитов и использование систем управления репозиториями, таких как GitHub, в том числе создание, клонирование, связывание и выгрузка репозиториев по SSH-ключу. Научились создавать, переключаться, сливать и удалять ветки, а также разрешать конфликты при слиянии. Познакомились с работой с ветками в удаленном репозитории и использованием опции --force для принудительной выгрузки изменений. Кроме того, мы провели форк репозитория и склонировали его для дальнейшей работы. Полученный опыт работы с Git и системами управления репозиториями будет полезен в будущих проектах.