

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»
Інститут прикладного системного аналізу
Кафедра системного проєктування

Лабораторна робота №1
з курсу *«Проектування інформаційних систем»*

Виконала:
студентка групи ДА-71
Опрісник Роксолана
Романівна

Київ — 2020

Мета роботи: за допомогою системи контролю версій завантажити коди програми у репозиторій. Відтворити типовий цикл розробки програмного забезпечення з використанням системи контролю версій.

Хід роботи

1. Для виконання лабораторних робіт було вибрано та встановлено клієнтське безкоштовне програмне забезпечення для роботи з системою контролю версій Github.

2. Опис команд, які використовувалися протягом виконання роботи з системою контролю версіями:

- Клонування сховища здійснюється командою **git clone [url]**.
- Визначення стану файлів:

Основний інструмент, який використовується для визначення, які файли в якому стані знаходяться - це команда **git status**. Якщо ви виконаєте цю команду відразу після клонування, ви побачите щось на зразок цього:

```
[user@MacBook-Pro-USER da71 % git status
On branch oprisnykr
nothing to commit, working tree clean
user@MacBook-Pro-USER da71 %
```

- Перегляд історії комітів
git log

За замовчуванням, без аргументів, git log виводить список комітів створених в даному репозиторії в зворотному хронологічному порядку. Тобто найостанніші комітів показуються першими. Як ви можете бачити, ця команда відображає кожен Коміт разом з його контрольної сумою SHA-1, ім'ям та електронною поштою автора, датою створення та коментарем.

- Індексація змінених файлів

Давайте модифікуємо файл, що знаходився під версійність контролем. Якщо ми змінимо файл,що відстежувався і після цього знову виконаємо команду status, то результат буде приблизно таким:

```
[user@MacBook-Pro-USER da71 % git status
On branch oprisnykr
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .DS_Store
        Oprisnyk/.DS_Store
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

Файл DS_Store знаходиться в секції "Changes not staged for commit" - це означає, що відстежується файл був змінений в робочому каталозі, але поки не проіндексовані. Щоб проіндексувати його, необхідно виконати команду git add (це багатофункціональна команда, вона використовується для додавання під версійність контроль нових файлів, для індексації змін, а також для інших цілей, наприклад для вказівки файлів з виправленим конфліктом злиття). Виконаємо git add, щоб проіндексувати DS_Store а потім знову виконаємо git

```
status:
user@MacBook-Pro-USER da71 % git add .
user@MacBook-Pro-USER da71 % git status
On branch oprisnykr
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .DS_Store
    new file:   Oprisnyk/.DS_Store
```

Тепер обидва файли проіндексовані і увійдуть в наступний коміт.

git checkout -b name створює нову гілку і переключає на неї

git commit -m " записує зміни із заданим повідомленням

git push -f origin branch-name відправлення змін у віддалений репозиторій не зважаючи на помилки

Перейдемо до консолі й виконаємо наступні команди:

1. Клонуємо репозиторій на власний комп'ютер.

```
git clone https://github.com/TarasVolovodenko/da71
```

2. Створимо власну гілку для роботи та перейдемо до неї

```
git checkout -b oprisnykr
```

3. Створимо в локальному репозиторії необхідні файли (папку і файл readme).

```
mkdir Oprisnyk
```

```
echo «#Initial commit»>Oprisnyk/README.md
```

4. Додамо їх до системи контролю версій.

```
git add Oprisnyk/README.md
```

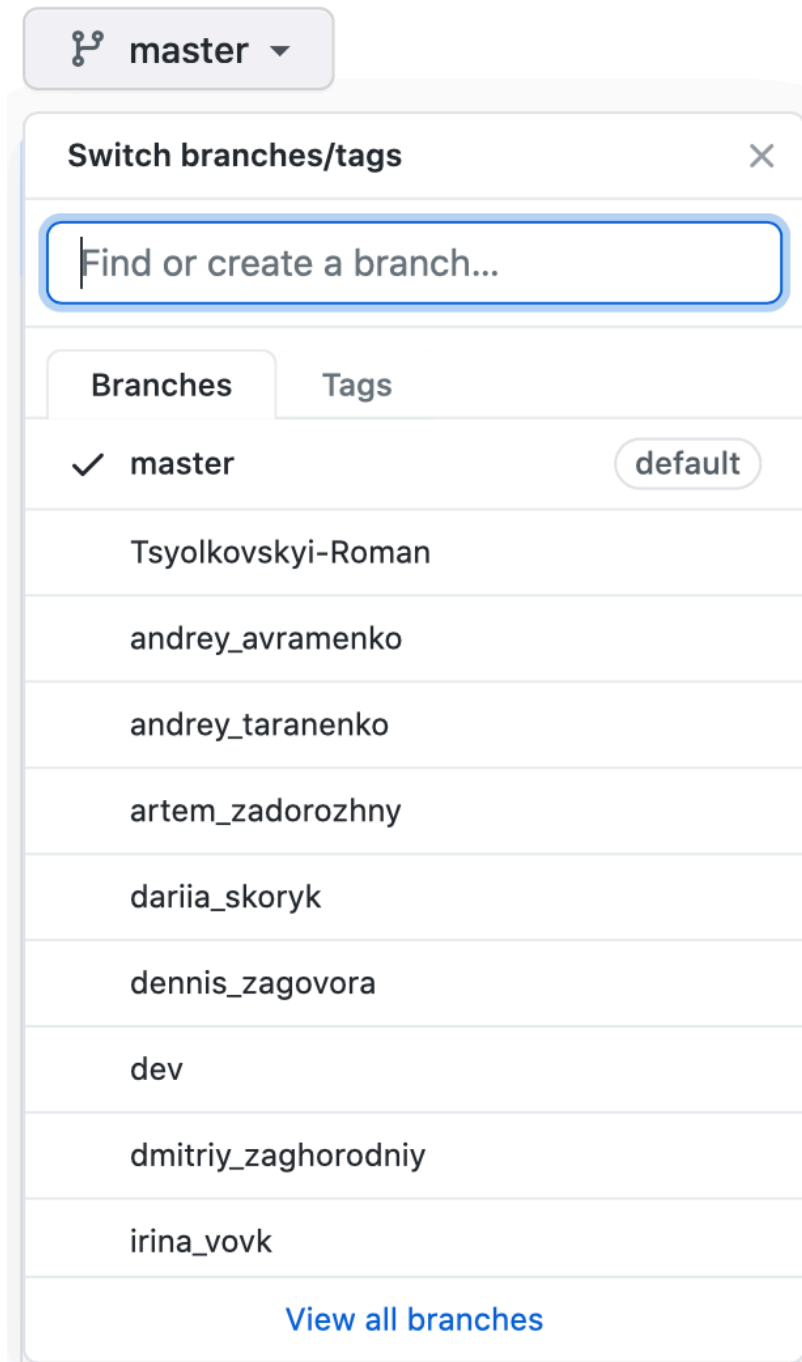
5. Створюємо «коміт»

```
git commit -m "Add README.md example"
```

6. Відправляємо «коміт» на сервер

```
git push -f origin oprisnykr
```

7. Заходимо на GitHub та переходимо на гілку master



8. Створюємо pull request і мерджимо гілку з master


[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [...](#)

[Edit](#) [Jump to bottom](#)


Add README.md example #23



Merged TarasVolovodenko merged 1 commit into `master` from `oprisnykr` 3 hours ago



[Conversation](#) 0 [Commits](#) 1 [Checks](#) 0 [Files changed](#) 1

 **oprisnykr** commented 3 hours ago Collaborator [...](#)

No description provided.



  Add README.md example de4b455

  **TarasVolovodenko** merged commit **ae488d4** into `master` 3 hours ago [Revert](#)

Pull request successfully merged and closed
You're all set—the `oprisnykr` branch can be safely deleted.

[Delete branch](#)

Висновки: В даній лабораторній роботі відтворено типовий цикл розробки програмного забезпечення з використанням системи контролю версій, продемонстровано та описано базові операції із системою контролю версій GIT, створено індивідуальну гілку для роботи, до неї додані файли роботи.