

Міністерство освіти і науки України  
Національний Технічний Університет України  
«Київський Політехнічний Інститут»  
Навчально-науковий комплекс  
«Інститут прикладного системного аналізу»  
Кафедра системного проектування

Лабораторна робота №1  
з дисципліни  
«Проектування інформаційних систем»  
Система контролю версій GIT.

Виконала:  
студентка групи ДА-72  
Потапова С. С.  
Варіант 24

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.

**Задача:**

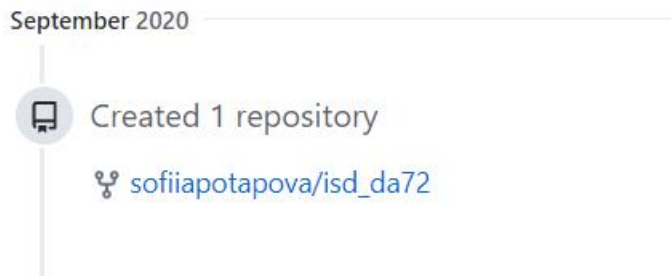
1. Вивчити основні команди роботи з репозиторіями.
2. Завантажити код програми у репозиторій.
3. Показати основний цикл роботи з програмним кодом за допомогою системи контролю версій.

**Завдання:**

1. Обрати безкоштовну систему репозиторія для системи контролю версіями, наприклад projectlocker, або інші.
2. Встановити клієнтське безкоштовне програмне забезпечення для роботи з системою контролю версій (GIT, SVN clients).
3. Протягом роботи над лабораторними роботами 2-6 використовувати систему контролю версіями.
4. Описати цикл розробки програмного забезпечення з використанням системи контролю версій.

## Хід роботи

1. Клонуємо свою персональну версію репозиторію за допомогою fork.



2. Клонуємо даний репозиторій собі на комп'ютер. Для цього переходимо у потрібну директорію і прописуємо команду git clone.

```
MINGW64:/c/Users/Мой компьютер/Desktop/4course/ПИС

Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~
$ cd Desktop/4course/ПИС

Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~/Desktop/4course/ПИС
$ git clone git@github.com:sofiapotapova/isd_da72
Cloning into 'isd_da72'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 15 (delta 2), reused 7 (delta 1), pack-reused 0
Receiving objects: 100% (15/15), done.
Resolving deltas: 100% (2/2), done.
```

3. Перейдемо у директорію проекту для роботи з ним. Створимо нову гілку і перейдемо на неї за допомогою команди git checkout.

```
Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~/Desktop/4course/ПИС
$ cd isd_da72

Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~/Desktop/4course/ПИС/isd_da72 (master)
$ git branch
* master

Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~/Desktop/4course/ПИС/isd_da72 (master)
$ git checkout -b dev
Switched to a new branch 'dev'

Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~/Desktop/4course/ПИС/isd_da72 (dev)
$ |
```

4. Перейдемо до папки з моїм прізвищем та відредагуємо README файл додавши у нього назву майбутнього проекту за допомогою команди echo “потрібний текст” > “потрібний файл”.

git status - показує поточний стан на гілці.

```
$ git status
On branch dev
nothing to commit, working tree clean

Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~/Desktop/4course/ПИС/isd_da72/Potapova_Sofia (dev)
$ ls
README.md

Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~/Desktop/4course/ПИС/isd_da72/Potapova_Sofia (dev)
$ echo "Task Manager" > README.md

Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~/Desktop/4course/ПИС/isd_da72/Potapova_Sofia (dev)
$ git status
On branch dev
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

5. Додамо файл у набір змін і зробимо коміт відповідно командами git add та git commit.

```
Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~/Desktop/4course/ПИС/isd_da72/Potapova_Sofia (dev)
$ git add README.md
warning: LF will be replaced by CRLF in Potapova_Sofia/README.md.
The file will have its original line endings in your working directory

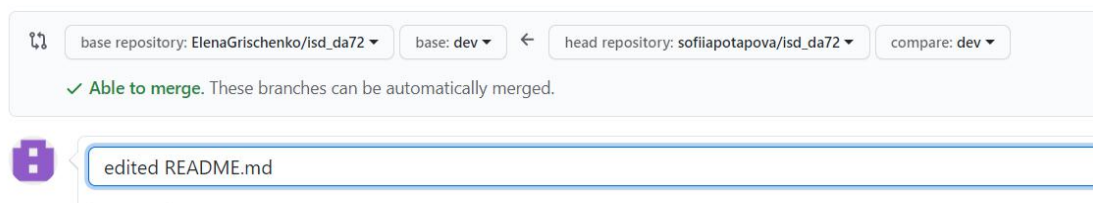
Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~/Desktop/4course/ПИС/isd_da72/Potapova_Sofia (dev)
$ git commit -m "edited README.md"
[dev 6f32e5c] edited README.md
1 file changed, 1 insertion(+)

Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~/Desktop/4course/ПИС/isd_da72/Potapova_Sofia (dev)
$
```

6. Відправимо всі зміни на гіт хаб командою `git push` (але при цьому так як ми створили на комп'ютері нову гілку, то одразу ж створимо і на віддаленому репозиторії цю гілку).

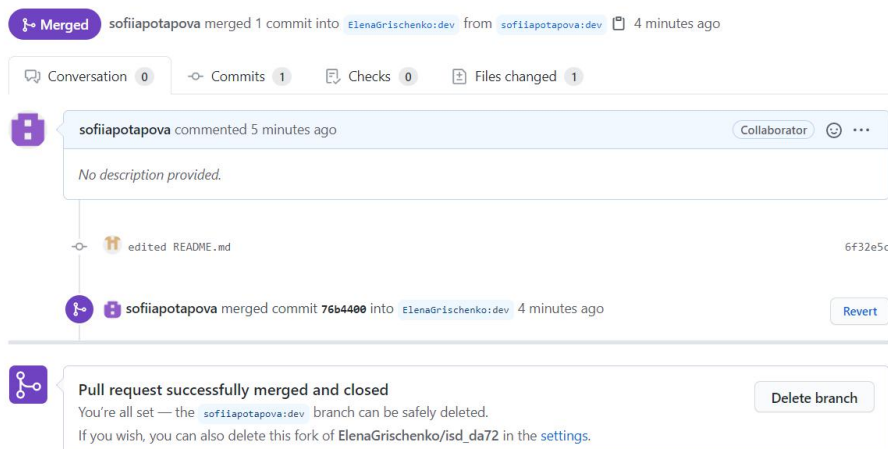
```
Мой компьютер@DESKTOP-VUUTDR5 MINGW64 ~/Desktop/4course/ПИС/isd_da72/Potapova_So
fiia (dev)
$ git push -u origin dev
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 331 bytes | 110.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'dev' on GitHub by visiting:
remote:   https://github.com/sofiapotapova/isd_da72/pull/new/dev
remote:
To github.com:sofiapotapova/isd_da72
 * [new branch]      dev -> dev
Branch 'dev' set up to track remote branch 'dev' from 'origin'.
```

7. Створюємо pull request.



8. Робимо Merge.

edited README.md #22



## **Висновки**

У даній лабораторній роботі ми розглянули роботу із системою контролю версій Git. Також використали на практиці основні команди по редагуванню, додаванню та створенню коміта, тобто відтворили ЖЦ ПО з використанням системи контролю версій. У ході виконання ніяких проблем не виникало.