Національний університет "Львівська політехніка" Інститут комп'ютерних наук та інформаційних технологій Кафедра систем автоматизованого проектування



До лабораторної роботи №1

З дисципліни "Управління ІТ-проектами"

На тему "Шаблон проектування МVC"

Виконав:

студ. групи КН-408 Пагута В.О.

Прийняв:

Василишин Б. С.

Мета роботи: ознайомитись з системами контролю версій. Зрозуміти принципи синхронізації робочих груп у гнучких командах. Набути навичок використання технічних засобів та протоколів для роботи з системами контролю версій.

Короткі теоретичні відомості

GIT Git (вимов. «гіт») – розподілена система керування версіями файлів. Проект був створений Лінусом Торвальдсом для управління розробкою ядра Linux. На сьогоднішній день підтримується Джуном Хама. Система спроектована як набір програм, спеціально розроблених з урахуванням їх використання в скриптах. Це дозволяє зручно створювати спеціалізовані системи контролю версій на базі Git або користувацькі інтерфейси. Наприклад, Cogito ϵ саме таким прикладом фронтенда до репозиторіїв Git, а StGit використовує Git для управління колекцією патчів . 9 Git підтримує швидке розділення і злиття версій, включає інструменти для візуалізації та навігації по нелінійній історії розробки. Git надає кожному розробнику локальну копію всієї історії розробки; зміни копіюються з одного репозиторію в іншій. Віддалений доступ до репозиторіїв Git забезпечується git-daemon, gitosis, SSH- або HTTPсервером. TCP-сервіс git-daemon входить в дистрибутив Git і є разом з SSH найбільш поширеним і надійним протоколом доступу. Метод доступу по НТТР, незважаючи на ряд обмежень, дуже популярний в контрольованих мережах, тому що дозволяє використовувати існуючі конфігурації мережевих фільтрів.

Лабораторне завдання

- 1. Ознайомитись з принципами роботи систем контролю версій.
- 2. Отримати індивідуальне завдання у викладача.
- 3. Написати програму згідно з індивідуальним завданням.
- 4. Створити репозиторій та завантажити туди свій програмний код.
- 5. Внести зміни в програму, при необхідності провести вирішення конфліктів.

Вимоги до програми

7. Програму, яка виконує частотний аналіз тексту: визначає усі слова, які зустрічаються в ньому і кількість їх входження. Результати виводить у вікно графічного інтерфейсу.

Реалізація програми

```
import random
import string
def replace_odd_digits_with_random_letter(text):
   result = []
   for c in text:
        if c.isdigit() and int(c) % 2 != 0:
            random_letter = random.choice(string.ascii_letters)
           result.append(random_letter)
        else:
           result.append(c)
   return ''.join(result)
def count_digits(text):
   return sum(c.isdigit() for c in text)
def count_digits_in_file_and_replace_odds(file_path):
   try:
       with open(file_path, 'r', encoding='utf-8') as file:
            text = file.read()
        # Рахуємо кількість цифр до зміни
        digit_count_before = count_digits(text)
        # Замінюємо непарні числа на випадкові літери
        modified_text = replace_odd_digits_with_random_letter(text)
        # Рахуємо кількість цифр після зміни
        digit_count_after = count_digits(modified_text)
        # Записуємо змінений текст назад у файл
        with open(file_path, 'w', encoding='utf-8') as file:
            file.write(modified text)
```

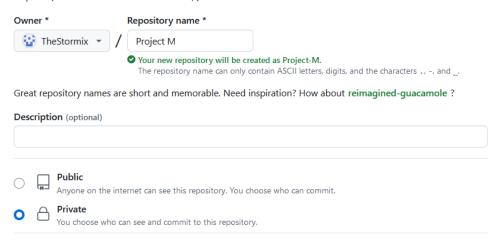
```
return digit_count_before, digit_count_after
except FileNotFoundError:
    print(f"Файл {file_path} не знайдено.")
    return 0, 0

# Вказати шлях до вашого файлу
file_path = 'text.txt'

digit_count_before, digit_count_after =
count_digits_in_file_and_replace_odds(file_path)
print(f"Кількість цифр у файлі до змін: {digit_count_before}")
print(f"Кількість цифр у файлі після змін: {digit_count_after}")
```

Результат виконання курс, у Кількість цифр у файлі до змін: 6 Кількість цифр у файлі після змін: 6

Після чого створюю GitHub репозиторій



Ініціалізую локальний репозиторій

C:\Users\pagut\OneDrive\Poбочий стіл\UNI\4 курс\управління\lab1>git init Initialized empty Git repository in C:/Users/pagut/OneDrive/Poбочий стіл/UNI/4 курс/ управління/lab1/.git/

Створюю файл та роблю перший коміт.

C:\Users\pagut\OneDrive\Pобочий стіл\UNI\4 курс\управління\lab1>git add text.txt
C:\Users\pagut\OneDrive\Pобочий стіл\UNI\4 курс\управління\lab1>git status

Виконую команду push на віддалений репозиторій.

```
C:\Users\pagut\OneDrive\Po6oчий crin\UNI\4 курс\управління\lab1>git commit -m "first commit"

[main (root-commit) 53de2a7] first commit

1 file changed, 1 insertion(+)
create mode 100644 text.txt

C:\Users\pagut\OneDrive\Po6oчий crin\UNI\4 курс\управління\lab1>git push -u project main
fatal: 'project' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

C:\Users\pagut\OneDrive\Po6oчий crin\UNI\4 курс\yправління\lab1>git remote add project https://github.com/TheStormix/Project-M.g

C:\Users\pagut\OneDrive\Po6oчий crin\UNI\4 курс\yправління\lab1>git push -u project main
Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Writing objects: 100% (3/3), 255 bytes | 255.00 KiB/s, done.

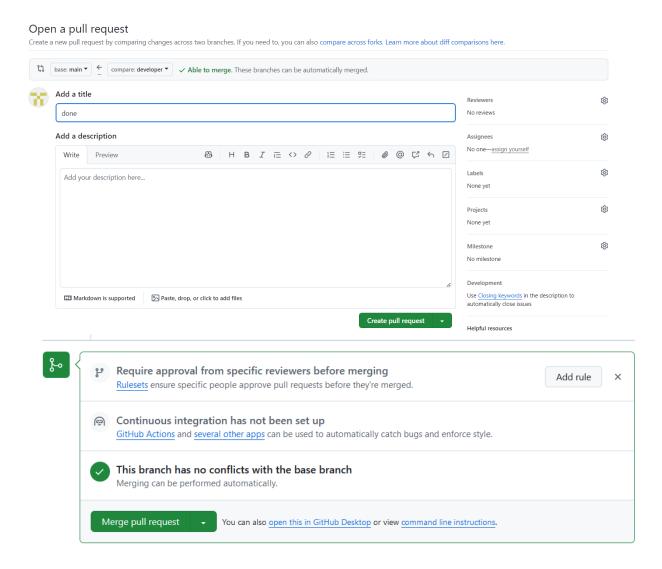
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

To https://github.com/TheStormix/Project-M.git
* [new branch] main -> main
branch 'main' set up to track 'project/main'.
```

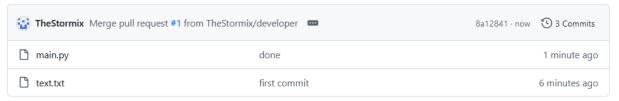
Створюю нову гілку "developer" та комічу туди свій код і пушу його на віддалений репозиторій.

```
C:\Users\pagut\OneDrive\Робочий стіл\UNI\4 курс\управління\lab1>git checkout -b developer
 Switched to a new branch 'developer'
 C:\Users\pagut\OneDrive\Pобочий стіл\UNI\4 курс\управління\lab1>git add main.py
 C:\Users\pagut\OneDrive\Pобочий стіл\UNI\4 курс\управління\lab1>git add text.txt
 C:\Users\pagut\OneDrive\Pобочий стіл\UNI\4 курс\управління\lab1>git status
 On branch developer
 Changes to be committed:
   (use "git restore --staged <file>..." to unstage)
    new file: main.py
C:\Users\pagut\OneDrive\Poбочий стіл\UNI\4 курс\управління\lab1>git commit -m "done"
[developer 518a517] done
 1 file changed, 45 insertions(+)
 create mode 100644 main.py
C:\Users\pagut\OneDrive\Poбочий стіл\UNI\4 курс\управління\lab1>git push project developer
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 936 bytes | 936.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'developer' on GitHub by visiting:
remote:
               https://github.com/TheStormix/Project-M/pull/new/developer
remote:
To https://github.com/TheStormix/Project-M.git
 * [new branch]
                        developer -> developer
C:\Users\pagut\OneDrive\Робочий стіл\UNI\4 курс\управління\lab1>
```

Створюю пул реквест для того, щоб змерджити зміни в гілку таіп.



Після вдалого мерджа отримав оновлену гілку таіп



Висновок: Отже, на даній лабораторній роботі я ознайомився з системами контролю версій. Зрозумів принципи синхронізації робочих груп у гнучких командах. Набув навичок використання технічних засобів та протоколів для роботи з системами контролю версій.