

Національний університет “Львівська політехніка”
Інститут комп’ютерних наук та інформаційних технологій
Кафедра систем автоматизованого проектування



Звіт

До лабораторної роботи №1
З дисципліни “Управління ІТ-проектами”
На тему “Шаблон проектування MVC”

Виконав:

студ. групи КН-408

Пагута В.О.

Прийняв:

Василишин Б. С.

Львів – 2024

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 є 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 інший. 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 HTTP, 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

```

Кількість цифр у файлі до змін: 6
Кількість цифр у файлі після змін: 6

```


Після чого створюю GitHub репозиторій


Owner *  TheStormix / Repository name *

✓ Your new repository will be created as Project-M.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

Great repository names are short and memorable. Need inspiration? How about [reimagined-guacamole](#) ?

Description (optional)

☐  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☒  **Private**
You choose who can see and commit to this repository.

Ініціалізую локальний репозиторій

```

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

```

Створюю файл та роблю перший коміт.

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

C:\Users\pagut\OneDrive\Робочий стіл\UNI\4 курс\управління\lab1>git status
On branch main
```

Виконую команду push на віддалений репозиторій.

```
C:\Users\pagut\OneDrive\Робочий стіл\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\Робочий стіл\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\Робочий стіл\UNI\4 курс\управління\lab1>git remote add project https://github.com/TheStormix/Project-M.git

C:\Users\pagut\OneDrive\Робочий стіл\UNI\4 курс\управління\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\Робочий стіл\UNI\4 курс\управління\lab1>git add main.py

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

C:\Users\pagut\OneDrive\Робочий 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\Робочий 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\Робочий 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 main.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#). [Learn more about diff comparisons here](#).

base: main ← compare: developer ✓ Able to merge. These branches can be automatically merged.

Add a title

done

Add a description

Write

Preview

Add your description here...

Markdown is supported

Paste, drop, or click to add files

Create pull request

Reviewers

No reviews

Assignees

No one—[assign yourself](#)

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Use [Closing keywords](#) in the description to automatically close issues

Helpful resources

Require approval from specific reviewers before merging

[Rulesets](#) ensure specific people approve pull requests before they're merged.

Continuous integration has not been set up

[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.




This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Після вдалого мерджа отримав оновлену гілку main

 TheStormix	Merge pull request #1 from TheStormix/developer	8a12841 · now	🕒 3 Commits
 main.py	done	1 minute ago	
 text.txt	first commit	6 minutes ago	

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.