Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №1

Технології розроблення програмного забезпечення

«*Системи контроля версій. Розподілена система контролю версій «Git».*»

Виконав
студент групи ІА–32:
Варивода К. С.

Перевірив:

Мягкий Михайло Юрійович

Київ 2025

Зміст

Теоретичні відомості

Git - є системою розподіленого контролю версій, коли кожен розробник має власний репозиторій, куди він вносить зміни. Далі система git синхронізує репозиторії із центральним репозиторієм. Це дозволяє проводити роботу незалежно від центрального репозиторію, перекладає складності ведення гілок та склеювання змін більше на плечі системи, ніж розробників.

Робота з гілками

При створенні репозиторію автоматично створюється головна гілка «master» або «main», тобто щоб створити головну гілку потрібно створити сам репозиторій:

- git init

Розробникам часто потрібно використовувати різні гілки для багатьох причин, від нових фіч до баг фіксів.

Для створення гілки є 3 варіанти:

1. git branch branch1 – створює гілку branch1, але не переходить у неї.
2. git checkout -b branch2 – створює гілку branch2 і одразу переходить.
3. git switch -c branch3 – сучасна команда для створення і переходу.

Також за потреби можна перевірити всі гілки в проекті:

- git branch

Або поточну:

- git status

Після створення гілки розробник має почати роботу в цій гілці, перехід на неї робиться через команди:

- git checkout <branch>
- git switch <branch>

В різних гілках можуть знаходитися різні файли. Це дає можливість паралельно розробляти окремі частини проекту

Після того, як фіча зроблена, а баг пофікшений ці оновлення потрібно застосувати на головній гілці. Це робиться шляхом зливання гілок.

1. Перейти на гілку в яку потрібно злити іншу гілку:
    - git checkout <branch>
2. Злити потрібну гілку:
    - git merge <branch>
3. При конфілікті - виправити конфлікт в файлах та
    - git merge –continue

    або

    - git merge –abort

Створення комітів

Git, для створення комітів, має спочатку проіндексувати файли, тобто додати додати змінені файли. Це робиться командою:

- git add <file>

Після того як файл було проіндексовано можна переходити до самого коміту, використовуючи:

- git commit –m “<comment>”

Також git дозволяє створювати пусті коміти, без файлів:

- git commit --allow-empty -m "Initial commit"

Хід роботи

1. **Створення папки та git репозиторію**

mkdir lab1

cd lab1

git init

2. **Перевіряємо статус репозиторію, перевіряємо на якій ми гілці**

git status

3. **Створюємо 3 гілки різними способами**

git branch branch1

git checkout -b branch2

git checkout master // Переходимо назад на головну гілку

git switch -c branch3

git checkout master

4. Створення 3 файлів з відповідним текстом

echo 1 > 1.txt

echo 2 > 2.txt

echo 3 > 3.txt

5. Додаємо файли до кожної з гілок

Гілка 1:

git checkout branch1

git add 1.txt

git commit –m “1.txt”

Гілка 2:

git checkout branch2

git add 2.txt

git commit –m “2.txt”

Гілка 3:

git checkout branch3

git add 3.txt

git commit –m “3.txt”

6. Додаємо файл 2.txt в гілку 1

git checkout branch1

echo b > 2.txt

git add 2.txt

git commit –m “2.txt”

7. Зливаємо branch2 у branch1

git merge branch2

Утворився конфлікт у файлі 2.txt

Вручну вирішуємо проблему:

nano 2.txt

Міняємо на потрібний текст

Зберігаємо ctrl + x, enter

git add 2.txt

git commit –m “2.txt”

git merge --continue

8. Перевірка історії комітів

git log –graph

Граф показує всі коміти з гілками і злиттям, включно з вирішенням конфліктів.

Висновки

Під час виконання лабораторної роботи, я навчився працювати с системою контролю версій git.