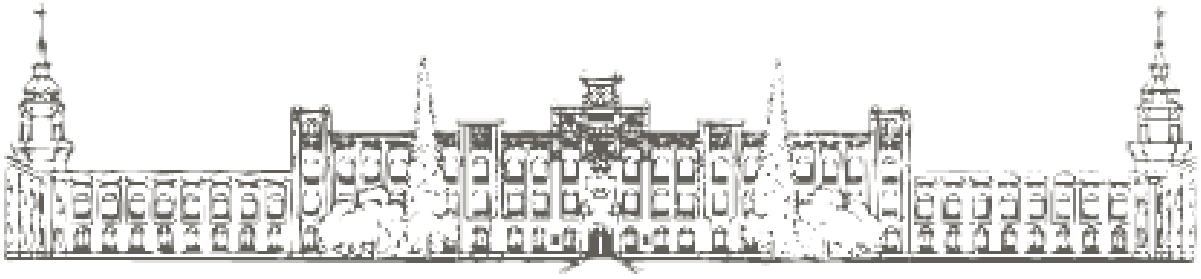

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Звіт лабораторної роботи №1

з курсу

«Технології розроблення програмного забезпечення»

Виконавець:
Нікітченко Наталя Олегівна
студентка групи ІА-33
залікова книжка № ІА-3318

«12» 09 2025 р.

Перевірив: **Мягкий М. Ю.**

Київ – 2025

1. ЛАБОРАТОРНА РОБОТА № 1

Тема: Системи контроля версій. Розподілена система контролю версій «Git».

Мета: Навчитися виконувати основні операції в роботі з децентралізованими системами контролю версій на прикладі роботи з сучасною системою Git.

Завдання:

• Ознайомитись із короткими теоретичними відомостями.

• Створити Git репозиторій.

• Клонувати Git репозиторій.

• Продемонструвати базову роботу з репозиторієм: створення версій, додавання тегів, робіт з гілками (створення та злиття), робота з комітами, вирішення конфліктів, а також робота з віддаленим репозиторієм.

Теоретичні відомості:

1.1 Призначення систем управління версіями

Система управління версіями (від англ. Version Control System або Source Control System) – програмне забезпечення яке призначено допомогти команді розробників керувати змінами в вихідному коді під час роботи [1]. Система керування версіями дозволяє додавати зміни в файлах в репозиторій і таким чином після кожної фіксації змін мани нову ревізію файлів. Це дозволяє повертатися до попередніх версій коду для аналізу внесених змін або пошуку, які зміни привели до появи помилки. Таким чином можна знайти хто, коли і які зміни зробив в коді, а також чому ці зміни були зроблені.

Такі системи найбільш широко використовуються при розробці програмного забезпечення для зберігання вихідних кодів програми, що розробляється. Однак вони можуть з успіхом застосовуватися і в інших областях, в яких ведеться робота з великою кількістю електронних документів, що безперервно змінюються. Зокрема, системи керування

версіями застосовуються 8 у САПР, зазвичай у складі систем керування даними про виріб (PDM). Керування версіями використовується у інструментах конфігураційного керування (Software Configuration Management Tools).

1.2 Робота з Git

Робота з Git може виконуватися з командного рядка, а також за допомогою візуальних оболонок. Командний рядок використовується програмістами, як можливість виконання всіх доступних команд, а також можливості складання складних макросів. Візуальні оболонки як правило дають більш наглядне представлення репозиторію у вигляді дерева, та більш зручний спосіб роботи з репозиторієм, але, дуже часто доступний не весь набір команд Git, а лише саме ті, що найчастіше використовуються.

Прикладами візуальних оболонок для роботи з Git є Git Extension, SourceTree, GitKraken, GitHub Desktop та інші.

Основна ідея Git, як і будь-якої іншої розподіленої системи контроля версій – кожен розробник має власний репозиторій, куди складаються зміни (версії) файлів, та синхронізація між розробниками виконується за допомогою синхронізації репозиторіїв. Процес роботи виглядає так, як зображено на рисунку 1.1.

Відповідно, є ряд основних команд для роботи [2]:

1. Клонувати репозиторій (git clone) – отримати копію репозиторію на локальну машину для подальшої роботи з ним;

2. Синхронізація репозиторіїв (git fetch або git pull) – отримання змін із віддаленого (вихідного, центрального, або будь-якого іншого такого ж) репозиторію;

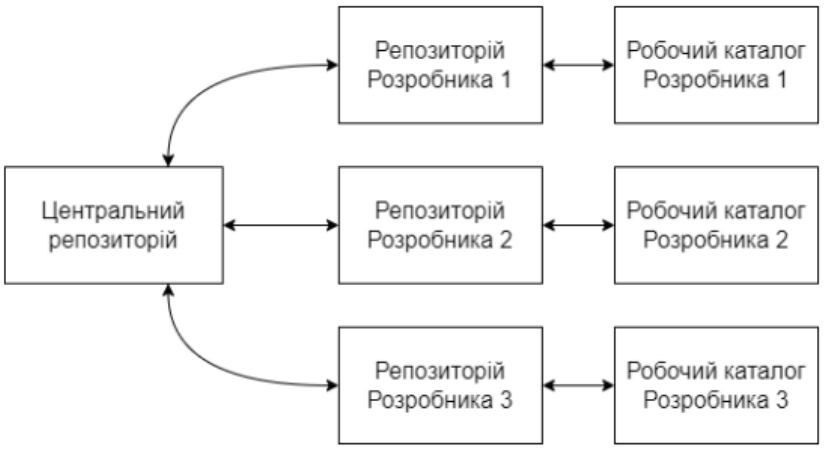

Рисунок 1.1. Схема процесу роботи з Git

3. Фіксація змін в репозиторій (git commit) – фіксація виконаних змін в програмному коді в локальний репозиторій розробника;

4. Синхронізація репозиторіїв (git push) – переслати зміни – push – передача власних змін до віддаленого репозиторію – Записати зміни – commit – створення нової версії;

5. Оновитись до версії – update – оновитись до певної версії, що є у репозиторії.

6. Об’єднання гілок (git merge) – об’єднання вказаною гілки в поточну (часто ще називається «злиттям»).

Таким чином, якщо розглядати основний робочий процес програміста в команді, то він виглядає наступним чином: На початку роботи з проектом виконується клонування, після цього, в рамках виконання поставленої задачі, створюється бранч і всі зміни в коді, зроблені в рамках цієї задачі фіксуються в репозиторії (періодично виконується синхронізація з основним репозиторієм). Далі, коли задача виконана, то виконується об’єднання гілки з основною гілкою і фінальна синхронізація з центральним репозиторієм.

Хід роботи:

1.Створити локальний репозиторій.

```
C:\Users\Наталя Нікітченко> git init lab1
Initialized empty Git repository in C:/Users/Наталя Нікітченко/lab1/.git/
```

2. Додати довільний файл з довільним текстом (в даному випадку текст – test).

```
C:\Users\Наталя Нікітченко>cd lab1

C:\Users\Наталя Нікітченко\lab1>echo "test"> hello.txt
```

3. Зафіксувати додавання файлу.

```
C:\Users\Наталя Нікітченко\lab1>git add hello.txt
```

```
C:\Users\Наталя Нікітченко\lab1>git commit -m "Hello added"
[master (root-commit) e1dd4ba] Hello added
 1 file changed, 1 insertion(+)
 create mode 100644 hello.txt
```

4. Додати нову директорію з довільним ім'ям.

```
C:\Users\Наталя Нікітченко\lab1> mkdir child_dir
```

5. Додати файл у директорію.

```
C:\Users\Наталя Нікітченко\lab1>cd child_dir

C:\Users\Наталя Нікітченко\lab1\child_dir> echo "inner"> inner.txt
```

6. Зафіксувати додавання директорії із файлом.

```
C:\Users\Наталя Нікітченко\lab1\child_dir>cd ..

C:\Users\Наталя Нікітченко\lab1>git add child_dir

C:\Users\Наталя Нікітченко\lab1>git commit -m "Add child directory with content"
[master 5c787dc] Add child directory with content
 1 file changed, 1 insertion(+)
 create mode 100644 child_dir/inner.txt
```

7. Створити гілку і перейти на неї.

```
C:\Users\Наталя Нікітченко\lab1>git branch feature_branch
```

```
C:\Users\Наталя Нікітченко\lab1>git checkout feature_branch
Switched to branch 'feature_branch'
```

8. Видалити додану директорію і зафіксувати зміни.

```
C:\Users\Наталя Нікітченко\lab1>rmdir /s child_dir
```

```
C:\Users\Наталя Нікітченко\lab1>git add .

C:\Users\Наталя Нікітченко\lab1>git commit -m "Remove child_dir"
[feature_branch dc3d355] Remove child_dir
 1 file changed, 1 deletion(-)
 delete mode 100644 child_dir/inner.txt
```

9. Злити зміни з основною гілкою.

```
C:\Users\Наталя Нікітченко\lab1>git checkout master
Switched to branch 'master'

C:\Users\Наталя Нікітченко\lab1>git merge feature_branch
Updating 5c787dc..dc3d355
Fast-forward
 child_dir/inner.txt | 1 -
 1 file changed, 1 deletion(-)
 delete mode 100644 child_dir/inner.txt
```

10. Вивести історію на екран.

```
C:\Users\Наталя Нікітченко\lab1>git log
commit dc3d3554ca529c447300054eea81236db58edcdd (HEAD -> master, feature_branch)
Author: niknat007 <nikitchenkonatalya27@gmail.com>
Date:   Fri Sep 12 21:28:49 2025 +0300

    Remove child_dir

commit 5c787dc5060e075da02a99816692a77a2eb2da07
Author: niknat007 <nikitchenkonatalya27@gmail.com>
Date:   Fri Sep 12 21:19:04 2025 +0300

    Add child directory with content

commit e1dd4ba57063624a001e2069aacccb7752abfe96
Author: niknat007 <nikitchenkonatalya27@gmail.com>
Date:   Fri Sep 12 21:15:40 2025 +0300

    Hello added

C:\Users\Наталя Нікітченко\lab1>
```

Висновки

В цій лабораторній роботі ми розглянули системи контроля версій та розподілену систему контролю версій «Git». А також розглянули та навчилися виконувати основні операції в роботі з децентралізованими системами контролю версій на прикладі роботи з сучасною системою Git.