Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 1

з дисципліни «Технології розроблення програмного забезпечення»

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

Виконала:
студентка групи ІА-33
Самойленко Анастасія

Перевірив:
ст. вик. кафедри ІСТ
М.Ю. Мягкий

Київ 2025

Вступ

Системи контролю версій є важливим інструментом у процесі розробки програмного забезпечення, оскільки вони дозволяють зберігати історію змін, організовувати командну роботу та забезпечувати надійність у керуванні проєктами. Використання таких систем дає змогу ефективно відстежувати зміни у файлах, повертатися до попередніх версій і синхронізувати роботу кількох розробників.

Однією з найпоширеніших сучасних систем контролю версій є **Git**. Це розподілена система, що надає кожному користувачу повну копію репозиторію з усією історією проєкту. Такий підхід робить роботу більш гнучкою, дозволяє працювати офлайн та легко інтегрувати зміни у спільний репозиторій. Git підтримує роботу з гілками, тегами, злиття змін та вирішення конфліктів, що робить його стандартом у програмуванні.

У межах даної лабораторної роботи ми повинні ознайомитися з основними можливостями системи Git, зокрема створенням і клонуванням репозиторіїв, виконанням базових операцій (коміти, робота з гілками, злиття, додавання тегів), а також використанням віддалених репозиторіїв для спільної роботи.

Теоретичні відомості

Призначення систем управління версіями

Системи управління версіями (Version Control Systems, VCS) — це програмні засоби, що дозволяють відстежувати та зберігати зміни у файлах, переважно у вихідному коді програм. Вони надають можливість створювати ревізії файлів, повертатися до попередніх версій, знаходити авторів змін та причини їх внесення. Такі системи широко застосовуються у розробці ПЗ, але також використовуються в інших сферах роботи з електронними документами.

Історія розвитку систем контролю версій

Розвиток систем контролю версій умовно поділяють на кілька етапів:

- Ранній етап – примітивне копіювання проєктів у різні папки; перша справжня система – RCS (1982), яка зберігала лише зміни у файлах.
- Централізовані системи (CVS, SVN) – з'явилися у 1990-х, використовували центральний сервер для спільної роботи. SVN стало більш надійною системою порівняно з CVS, однак мало проблеми з гілками та злиттями.
- Децентралізація (Git, Mercurial, ClearCase) – кожен розробник отримує повну копію репозиторію. Git (2005, Лінус Торвальдс) став революційною системою завдяки швидкості, гнучкості та зручній роботі з гілками. Mercurial був схожим конкурентом, але поступово втратив популярність.
- Хмарні платформи (GitHub, GitLab, Bitbucket) – з 2010-х років інтегрують Git з сервісами для командної роботи, CI/CD, DevOps, аналітики та автоматичного тестування.

Робота з Git

Git може використовуватися через:

- Командний рядок – забезпечує повний доступ до всіх команд і сценаріїв автоматизації.
- Графічні оболонки (GUI) – зручні для візуалізації репозиторію, але часто мають обмежений набір команд.

Основні можливості Git: створення локальних і віддалених репозиторіїв, виконання комітів, робота з гілками, злиття змін, розв'язання конфліктів, використання тегів та синхронізація з хмарними платформами.

Хід роботи

1. Для початку нам потрібно створити локальний репозиторій. Для цього ми відкриємо термінал та пропишемо команду ***git init lab1***(рис.1). У результаті з'явився новий каталог з початковою структурою репозиторію Git.

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

C:\Users\ANAST>
```

Рисунок 1 – Створення локального репозиторію

2. Далі нам треба створити і закоміти в репозиторії файл. Для цього ми додамо довільний файл з довільним текстом до репозиторію (рис.2)

```
C:\Users\ANAST>cd lab1

C:\Users\ANAST\lab1>echo "test" > readme.txt
```

Рисунок 2 – Створення файлу з текстом

Додамо файл у список відстеження та виконаємо коміт (рис.3). Як бачимо тепер у репозиторії є перша зафіксована версія проєкту.

```
C:\Users\ANAST\lab1>git add readme.txt

C:\Users\ANAST\lab1>git commit -m "Added the first readme.txt file"
[master (root-commit) 0baba4d] Added the first readme.txt file
 1 file changed, 1 insertion(+)
 create mode 100644 readme.txt
```

Рисунок 3 – Додавання файлу та його коміт

3. Створимо нову папку (директорію) ***src*** для зберігання вихідних файлів(рис.4). Додаємо зміни до репозиторію та виконуємо коміт. Зафіксувати додавання директорії із файлом.

```
C:\Users\ANAST\lab1>mkdir src

C:\Users\ANAST\lab1>echo "print('Hello, Git!')" > src/app.py

C:\Users\ANAST\lab1>git add src/

C:\Users\ANAST\lab1>git commit -m "Створено директорію src з файлом app.py"
[master d37ea8e] Створено директорію src з файлом app.py
 1 file changed, 1 insertion(+)
 create mode 100644 src/app.py
```

```
C:\Users\ANAST\lab1>cd src

C:\Users\ANAST\lab1\src>echo "inner" > read.txt

C:\Users\ANAST\lab1\src>cd ..

C:\Users\ANAST\lab1>git add src

C:\Users\ANAST\lab1>git commit -m "Add child directory with content"
[master bebfb76] Add child directory with content
 1 file changed, 1 insertion(+)
 create mode 100644 src/read.txt
```

Рисунок 4 – Створення нової директорії

4. Нам потрібно створити нові гілки в нашому проєкті. Для цього початково перевіремо вже існуючі гілки. (рис.5). Далі додамо нову гілку ***feature*** та перейдемо на неї (рис.6).

```
C:\Users\ANAST\lab1>git branch
* master
```

Рисунок 5 – Гілки проєкту

```
C:\Users\ANAST\lab1>git branch feature

C:\Users\ANAST\lab1>git checkout feature
Switched to branch 'feature'
```

Рисунок 6 – Створення та перехід на нову гілку

```
C:\Users\ANAST\lab1>echo "print('Changes in the feature branch')" >> src/app.py

C:\Users\ANAST\lab1>git add src/app.py

C:\Users\ANAST\lab1>git commit -m "Changes added to branch"
[feature c03e2c1] Changes added to branch
 1 file changed, 2 insertions(+)
```

Рисунок 7 – Внесення змін у гілку

5. Гілка зі змінами злита з основною гілкою (рис.8)

```
C:\Users\ANAST\lab1>git merge feature
```

```
C:\Users\ANAST\lab1>git commit -m "Merge branch 'feature' into master"
[master 0ae97b5] Merge branch 'feature' into master
```

Рисунок 8 – злиття гільки feature з головною гілкою

```
C:\Users\ANAST\lab1>git log --graph
*   commit 0ae97b581b3bf29d6a4e3e726f945c3fdec320a9 (HEAD -> master)
|\  Merge: bebfb76 e649ebe
| | Author: SamoilenkoAnastasia <samoilenko.anastasia@lll.kpi.ua>
| | Date:   Sat Sep 13 11:16:34 2025 +0300
| |
| |     Merge branch 'feature' into master
| |
| * commit e649ebe314bcaccce064e635e8370dbd69ee9f65 (feature)
| | Author: SamoilenkoAnastasia <samoilenko.anastasia@lll.kpi.ua>
| | Date:   Sat Sep 13 11:10:25 2025 +0300
| |
| |     Changes added to the feature branch
| |
| * commit c03e2c1ff005f284fc4647c99e123084f111f3a8
| | Author: SamoilenkoAnastasia <samoilenko.anastasia@lll.kpi.ua>
| | Date:   Sat Sep 13 10:51:20 2025 +0300
| |
| |     Changes added to branch
| |
* | commit bebfb767b5dba0bee577751784732dc8e0a11fc8
|/  Author: SamoilenkoAnastasia <samoilenko.anastasia@lll.kpi.ua>
|   Date:   Sat Sep 13 10:33:26 2025 +0300
|
|       Add child directory with content
|
* commit d37ea8ee71330f6ba8aa0202c7435baf5e59ceae
| Author: SamoilenkoAnastasia <samoilenko.anastasia@lll.kpi.ua>
| Date:   Sat Sep 13 10:20:34 2025 +0300
|
|     Створено директорію src з файлом app.py
|
* commit 0baba4d0b44bcd3bc0711558e08e0ebd3e8a73a3
  Author: SamoilenkoAnastasia <samoilenko.anastasia@lll.kpi.ua>
  Date:   Sat Sep 13 10:09:06 2025 +0300

      Added the first readme.txt file
```

Рисунок 9 -Перегляд історії комітів у вигляді графа

Висновки

У ході лабораторної роботи я ознайомилася з принципами роботи систем контролю версій та практично опанувала роботу з розподіленою системою Git. Я створила локальний репозиторій, додала файли та директорії, виконала коміти для фіксації змін. Створила додаткову гілку для внесення змін, успішно її злила з основною гілкою та завершила merge. Під час роботи я навчилася працювати з гілками, комітами та синхронізувати зміни, що дало практичні навички організації версійного контролю у проєктах.