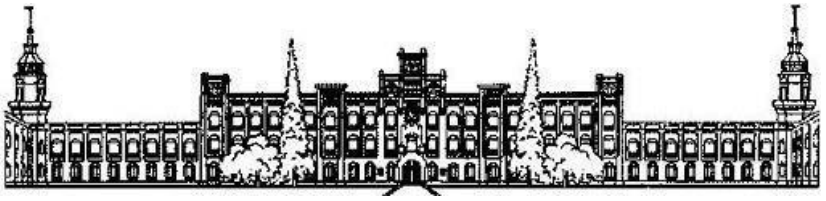

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 1

з дисципліни «Технології розроблення
програмного забезпечення»

Тема: «Git»

Виконав:
студент групи - ІА-32
Діденко Я.О

Перевірив:
Мягкий
М.Ю

Київ 2025

Теоретичні відомості

GitHub – це онлайн-платформа для зберігання та спільної роботи з Git-репозиторіями. Вона надає зручний веб-інтерфейс, додаткові інструменти та можливості командної роботи. Він використовується мільйонами розробників у всьому світі для ведення відкритих і закритих проєктів. Основна ідея полягає в тому, що будь-який репозиторій, який існує локально на комп'ютері, можна розмістити у віддаленому сховищі. Це дозволяє не тільки зберігати код у хмарі, а й робити його доступним для інших учасників команди або спільноти.

Окрім зручного зберігання, GitHub забезпечує:

- Механізм командної роботи – кілька людей можуть одночасно вносити зміни в різні частини проєкту, створювати нові гілки та об'єднувати їх.
- Керування доступом – власник може призначати ролі (читання, запис, адміністрування), робити репозиторій публічним або приватним.
- Прозорість розробки – завдяки історії комітів та інструментам для обговорень видно, хто і коли вніс певні зміни.
- Pull request (PR) – ключова функція для перевірки коду перед злиттям у головну гілку.

GitHub також підтримує інтеграції з іншими сервісами: системами автоматичного тестування, системами зборки та розгортання (CI/CD), таск-трекерами. Це робить платформу не лише сховищем коду, а й повноцінним середовищем для керування розробкою програмного забезпечення.

Основні поняття

- Віддалений репозиторій (remote) – копія проєкту, що зберігається на GitHub. Підключення: **git remote add origin <url>**
- Клонування (clone) – завантаження віддаленого репозиторію на локальний комп'ютер:
  git clone <url>
- Завантаження змін (push) – відправлення локальних комітів у віддалений репозиторій:
  git push origin branch_name
- Отримання змін (pull) – завантаження й об'єднання змін із віддаленого репозиторію:
  git pull origin branch_name
- Fork – створення власної копії чужого репозиторію для роботи з ним.
- Pull request (PR) – запит на внесення змін з однієї гілки або форку в інший репозиторій.

Хід роботи

1. Створення нового каталогу та ініціалізація Git репозиторію

mkdir 1
cd 1
git init

2. Створення порожнього коміту

git commit --allow-empty -m "init"

3. Створення трьох гілок різними способами

git checkout -b b1 # спосіб 1: створити і перейти
git checkout main
git branch b2 # спосіб 2: створити гілку без переходу
git switch -c b3 # спосіб 3: створити і перейти
git checkout main

4. Створення трьох файлів

touch f1.txt f2.txt f3.txt

5. Редагування файлів

nano
f1.txt
nano
f2.txt
nano
f3.txt

6. Розміщення файлів у відповідних гілках

git checkout b1
git add f1.txt
git commit -m "f1"

git checkout b2
git add f2.txt
git commit -m "f2"

git checkout b3
git add f3.txt
git commit -m "f3"

7. Додавання файлу f1.txt у гілку b3

echo "123" > f1.txt
git add f1.txt
git commit -m "f1"

8. Злиття гілки b1 у b3

git merge b1

Виник конфлікт у файлі f1.txt
(обидві гілки додали файл).

Конфлікт вирішуємо таким чином:

редагування файлу - видалення
службових рядків <<<<<<<,
=======, >>>>>>>.

Після цього:

git add f1.txt
git merge --continue

9. Перевірка історії комітів

git log --graph

Контрольні запитання

1. Що таке система контролю версій (СКВ)?
СКВ — це програма, яка зберігає історію змін у файлах, дозволяє відстежувати, хто і коли зробив зміни, повертатися до попередніх версій і працювати в команді.

2. Відмінності між розподіленою та централізованою СКВ.
Централізована (наприклад, SVN): єдиний центральний сервер, всі працюють напряму з ним. Якщо сервер недоступний — робота зупиняється.
Розподілена (наприклад, Git): кожен розробник має повну копію репозиторію з усією історією. Можна працювати офлайн, обмінюватися змінами через push/pull.

3. Різниця між stage та commit в Git.
Stage (staging area) — проміжна зона, куди додаються зміни (`git add`).
Commit — створення знімку (snapshot) з тих змін, що вже у staging.

4. Як створити гілку в Git?
```bash
git branch new-branch     # створити
git checkout -b new-branch # створити і одразу перейти
git switch -c new-branch   # аналог checkout
```

5. Як створити або скопіювати репозиторій Git з віддаленого серверу?
```bash
git clone https://github.com/user/repo.git
```

6. Що таке конфлікт злиття, як створити конфлікт, як вирішити конфлікт? Конфлікт злиття — ситуація, коли у різних гілках один і той самий рядок файлу змінено по-різному.
Створити: змінити один і той самий файл по-різному у двох гілках, потім зробити `git merge`.
Вирішити: відкрити файл, прибрати службові рядки `<<<<<<<`, `=======`, `>>>>>>>`, залишити правильний варіант → `git add` → `git commit` (або `git merge --continue`).

7. В яких ситуаціях використовуються merge, rebase, cherry-pick?
`merge` — для об’єднання гілок і збереження історії (наприклад, зливання feature у main).
`rebase` — для переписування історії та вирівнювання комітів (зручно для чистої лінійної історії).
`cherry-pick` — щоб перенести конкретний коміт з однієї гілки в іншу (без злиття всіх змін).

8. Як переглянути історію змін Git репозиторію в консолі?
```bash
git log
git log --oneline --graph --decorate --all
```
9. Як створити гілку в Git не використовуючи команду git branch?
```bash
git checkout -b new-branch
# або
git switch -c new-branch
```

10. Як підготувати всі зміни в поточній папці до коміту?
```bash
git add .
```

11. Як підготувати всі зміни в дочірній папці до коміту?
```bash
git add path/to/folder/
```

12. Як переглянути перелік наявних гілок в репозиторії?
```bash
git branch        # локальні гілки
git branch -a     # локальні + віддалені
```

13. Як видалити гілку?
```bash
git branch -d branchName            # видалити локально
git push origin --delete branchName   # видалити на сервері
```

14. Які є способи створення гілки та в чому між ними різниця?
`git branch name` — створює гілку, але не переходить у неї.
`git checkout -b name` — створює гілку і одразу переходить.
`git switch -c name` — сучасна альтернатива checkout, робить те саме.

Висновки

Git — це розподілена система контролю версій, яка дозволяє гнучко працювати з історією проекту, відстежувати зміни та працювати в команді.
Ключові поняття: stage, commit, branch, merge. Вміння працювати з гілками (створювати, зливати, вирішувати конфлікти) та історією (log, rebase, cherry-pick) є основними навичками для ефективної розробки.